

AD-A273 726

AFIT/GAE/ENY/93D-22



NONLINEAR LARGE DEFORMATION THEORY
OF COMPOSITE ARCHES USING
TRUNCATED ROTATIONS

THESIS

Daniel A. Miller II
Captain, USAF

AFIT/GAE/ENY/93D-22

93-30472-506



Approved for public release; distribution unlimited

93 12 1508 1

**Best
Available
Copy**

**NONLINEAR LARGE DEFORMATION THEORY
OF COMPOSITE ARCHES USING
TRUNCATED ROTATIONS**

THESIS

**Presented to the Faculty of the Graduate School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Aeronautical Engineering**

Daniel A. Miller II, B.S.M.E.

Captain, USAF

December 1993

Accesion For	
NTIS	CRA&I
DTIC	TAB
Unannounced	
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

Approved for public release; distribution unlimited

I wish to extend my greatest thanks to Dr Anthony Palazotto. As my thesis advisor, Dr Palazotto was always patient in explaining concepts again and again. I learned a great deal under his leadership and guidance. I would also like to thank Capt Scott Schimmels for providing the *MACSYMA* files that provided the frame work for implementing this research by forming element stiffness matrices. He also helped discover illusive kinematic inconsistencies. I would like to extend thanks to Dr Arnold Mayer WL/FIV and Dr Arje Nachman AFOSR for their sponsorship of this research. Finally, I would like to thank my family for their support during some busy and difficult times.

Table of Contents

List of Figures	v
List of Tables	viii
Abstract	ix
I. Introduction	1-1
1.1 Background	1-1
1.2 Literature Review	1-3
II. Theory	2-1
2.1 Kinematics	2-1
2.2 Constitutive Relations	2-11
2.3 Strain Relations.....	2-14
2.4 Beam Potential Energy	2-19
2.5 Finite Element Formulation	2-25
2.6 Numerical Solution Methods.....	2-30
2.7 Step by Step Riks Algorithm	2-35
2.8 Arch Geometry Definitions.....	2-37
III. Discussion and Results	3-1
3.1 Clamped Isotropic Shallow Thin Arch.....	3-1
3.2 Clamped Isotropic Shallow Thick Arch	3-3
3.3 Clamped Isotropic Thin Straight Beam.....	3-6
3.4 Cantilever Isotropic Thin Beam.....	3-8
3.5 Cantilever Isotropic Beam with Tip Moment	3-11
3.6 Cantilevered Composite Beam with End Load.....	3-14

3.7 Hinged Isotropic Shallow Thick Arch.....	3-17
3.8 Deep Hinged Isotropic Thin Arch.....	3-23
3.9 Hinged Clamped Isotropic Very Deep Arch #1.....	3-27
3.10 Hinged Clamped Very Deep Isotropic Arch #2.....	3-33
IV Vectorization and Parallelization Considerations.....	4-1
4.1 Vectorization	4-1
4.2 Parallel Processors	4-5
V. Conclusions and Recommendations	5-1
5.1 Summary and Conclusions	5-1
5.2 Recommendations for Further Work	5-3
Appendix A	A-1
A.1 Qij Transformations of Eqn(2-21).....	A-1
A.2 L, S and H Matrices in Eqns (2-52) and (2-54)	A-1
A.3 Element Stiffness Matrices of Eqn(2-56).....	A-5
A.4 Element Strain Energy Calculation	A-7
Appendix B. MACYSMA Inputs to Generate N1 and N2 Subroutines.....	B-1
B.1 N1 Inputs	B-1
B.2 N2 Inputs	B-6
Appendix C FORTRAN Program Description	C-1
C.1 Program Description.....	C-1
C.2 Data Input Format.....	C-3
C.3 FORTRAN Code Listing	C-6
Bibliography.....	BIB-1

List of Figures

Figure 2-1 Flat Beam Bending Rotation.....	2-2
Figure 2-2 Bending Motion	2-3
Figure 2-3 Global Coordinates of an Arch	2-5
Figure 2-4 Mid Plane Displacement Effects.....	2-6
Figure 2-5 Pure Bending Convention.....	2-7
Figure 2-6 Pure Shear Convention.....	2-9
Figure 2-7 Slope, Shear and Bending Relationships	2-10
Figure 2-8 Material and Global Coordinate Relationships.....	2-12
Figure 2-9 Beam Finite Element	2-26
Figure 2-10 General Load Displacement Curve.....	2-31
Figure 2-11 Displacement Control	2-32
Figure 2-12 Riks Technique.....	2-36
Figure 2-13 Typical Arch Geometry	2-37
Figure 3-1 Clamped Clamped Shallow Thin Arch	3-1
Figure 3-2 Comparison for Clamped Clamped Shallow Thin Arch	3-2
Figure 3-3 Clamped Clamped Shallow Thick Arch.....	3-4
Figure 3-4 Comparison for Clamped Clamped Shallow Thick Arch.....	3-5
Figure 3-5 Clamped Clamped Flat Thin Beam.....	3-6
Figure 3-6 Equilibrium Curve and Comparison for Clamped Clamped Thin Beam	3-7
Figure 3-7 Cantilever Isotropic Beam	3-8
Figure 3-8 End Displacement for Cantilever Isotropic Beam.....	3-10
Figure 3-9 Cantilevered Isotropic Beam with Tip Moment.....	3-11
Figure 3-10 Cantilever Beam with Tip Moment - Vertical Displacement.....	3-13

Figure 3-11 Cantilever Beam with Tip Moment - Horizontal Displacement.....	3-14
Figure 3-12 Cantilever Composite Beam.....	3-15
Figure 3-13 Vertical Displacement Comparison for Cantilever Composite Beam.....	3-16
Figure 3-14 Horizontal Displacement Comparison for Cantilever Composite Beam.....	3-16
Figure 3-15 Hinged Hinged Shallow Thick Arch.....	3-18
Figure 3-16 Equilibrium Path for Hinged Hinged Shallow Thick Arch.....	3-19
Figure 3-17 Displacement Control vs Riks Method.....	3-20
Figure 3-18 Hinged Hinged Shallow Arch Deformed Shapes	3-21
Figure 3-19 Hinged Hinged Deep Thin Arch.....	3-23
Figure 3-20 Load vs Vertical Displacement for Hinged Hinged Deep Arch.....	3-24
Figure 3-21 Deformed Shapes for Deep Hinged Hinged Arch	3-25
Figure 3-22 Shear Locking Deep Hinged Hinged Arch	3-26
Figure 3-23 Very Deep Hinged Clamped Arch #1.....	3-27
Figure 3-24 Mesh Refinement Very Deep Hinged Clamped Arch #1	3-28
Figure 3-25 Deformed Shapes Showing Element Kinking for Very Deep Hinged Clamped Arch	3-29
Figure 3-26 Strain Energy for Typical Elements.....	3-31
Figure 3-27 Very Deep Hinged Clamped Arch #2.....	3-34
Figure 3-28 Load Displacement for Very Deep Hinged Clamed Arch.....	3-36
Figure 3-29 Riks for Very Deep Hinged Clamped Arch	3-36

Figure 3-30 Deformed Shapes of Very Deep Hinged Clamped Arch	3-38
Figure 4-1 CPU Time Comparisons	4-5
Figure 4-2 Parallel Processor Domain Substructuring	4-7
Figure 4-3 Serial and Parallel Element Stiffness Processes.....	4-10

List of Tables

Abstract

This research has been directed toward capturing large cross sectional rotation during bending of composite arches. A potential energy based nonlinear finite element model that incorporates transverse shear strain was modified to include large bending rotations. Large rotation kinematics were derived in a vector format leading to nonlinear strain that was decomposed into convenient forms for inclusion in the potential energy function. Problem discretization resulted in a finite element model capable of large deformations. Riks method and displacement control solution techniques were used. Numerous problems and examples were compared and analyzed. Code vectorization and parallelization were briefly examined to improve computational efficiency.

**NONLINEAR LARGE DEFORMATION THEORY
OF COMPOSITE ARCHES USING
TRUNCATED ROTATIONS**

I. Introduction

1.1 Background

Today's aerospace industry has advanced to the point of using optimum design techniques in virtually all engineering applications. In structural elements, orthotropic fiber composite materials have emerged as lighter, stronger and many times, a more easily manufactured solution to a material application problem. Composites have the distinct advantage of being designed and built to many different specifications by varying materials, amount of fiber/matrix, and ply orientation. As with other high performance applications, the fiber composite analysis techniques are more complicated than their simpler isotropic counterparts.

The US Air Force uses thin composite shell structures in many existing systems and is likely to use them extensively in the future. Shells are defined as curved structures with one dimension being small compared to the others (thin). If another dimension is relatively small (width), then the geometry may be referred to as an arch. This research has been directed towards capturing large bending behavior of transversely loaded flat beams and curved arches with rectangular cross sections. Unlike other one dimensional

techniques that usually arise through beam theory, this thesis is derived from a more complicated two dimensional shell theory.

One problem of interest involves arches that experience large deformations, but small strains. This problem class is considered geometrically nonlinear, but allows the material properties to be modeled in a linear elastic manner. Structural stiffness changes come about from changes in the geometry during deformation and not from changes in the material properties due to plasticity, creep etc. Much work has been done to predict the behavior of anisotropic plates and shells. Palazotto and Dennis [27] have developed a geometrically nonlinear shell theory and accompanying FORTRAN code to trace the equilibrium path of orthotropic cylindrical shells. They include the following in their theory:

1. geometric nonlinearity with large displacement and moderate rotations,
2. linear elastic behavior of laminated anisotropic materials,
3. cylindrical shells and flat plates,
4. parabolic distribution of transverse shear stress,
5. and a finite element approach.

Creaghan [8] narrowed the class of problems considered by Palazotto and Dennis from two dimensional plates and shells to one dimensional beams and arches. He retained all of the features developed by Palazotto and Dennis, but reduced the theory by one dimension. Creaghan's research resulted in a relatively simple FORTRAN code that is easily modified with further theory enhancements. Creaghan demonstrated good problem solutions (as compared to other analytical solutions and test data) to 23 degrees of bending rotation. Beyond that point, significant solution divergence occurred for numerous beam and arch problems. The current research is directed to improving upon this rotation limit by incorporating a large bending rotation theory. Initially, Creaghan's code is retained as a framework. His simple approach and one dimensionality make it an excellent baseline to

build to. Before examining the current theory derivation, the author will briefly review the history of shell theory and examine what others have published in terms of beam and arch nonlinearity. The literature review will be concluded by discussing other large rotation theories and finally reviewing work that has been documented on advanced computing concepts for nonlinear finite element problems.

1.2 Literature Review

One of the earliest two dimensional theory designed to describe the three dimensional problem of thin flat plates was developed by Kirchhoff [30]. Kirchhoff assumed:

1. The middle plane remained unstrained (inextensible).
2. Through-the-thickness normal stress and strain are small compared to others and are neglected.
3. Cross section normals to the mid-plane remain normal throughout bending. There is no warping of the cross section. This assumption essentially neglects all transverse shear effects.

Kirchhoff's theory was extended to thin curved structures by Love [30]. The emerging theory, referred to as Kirchhoff-Love shell theory, also neglected transverse stress shear stress and strain. Koiter [19] showed that refinements of the Love theory are of little use unless transverse shear deformation effects are included.

Reissner [29] and Mindlin [21] added transverse shear to the Kirchhoff-Love development. The Reissner-Mindlin (RM) theory includes transverse shear that is constant through-the-thickness. This results in a cross section that can rotate, but not warp. RM theory does not satisfy the stress free boundary condition at the top and bottom surfaces of the shell. Consequently, when RM theory is used in a finite element

model, certain problems can experience shear locking. Shear locking occurs when the transverse shear causes the structure to act much stiffer than is physically correct. Shear correction factors are usually used to avoid locking when using a theory that incorporates shear but doesn't satisfy stress free boundary conditions. Elements that have nodes on the top and bottom surfaces with incorrect shear representation are likely to lock when the structure gets thin unless correction factors are applied.

Reddy [28] and others have developed theories with displacement functions that are cubic in the thickness coordinate. This results in a parabolic transverse shear distribution. With the proper choice of constants, these theories satisfy the stress free boundary conditions at the surfaces, thereby eliminating shear locking concerns. This parabolic shear distribution was used by Palazotto and Dennis [27] and Creaghan [8] and consequently is used here.

Beam and arch theories have been developed in a manner similar to shells and plates. Due to their one dimensional nature, most of these theories tend to be derived from beam theory and are consequently easier to implement than most 2-D theories. Although not all the theories reviewed are strictly 1-D, they are analyzed to determine their applicability to the beam and arch problems of interest. All of the mentioned authors analyze at least one beam or arch problem with the theory discussed. As with plates and shells, beams and arches have been analyzed using Kirchhoff-Love and Reissner-Mindlin approaches. Many theories include higher order shear representations. As we briefly review some other theories, the complexity of a certain method can be roughly evaluated by answering the following questions:

1. Is the formulation updated or total Lagrangian? Some updated Lagrangian theories lose accuracy as the geometry becomes very nonlinear. Total Lagrangian formulations always reference the original coordinate system while updated Lagrangian formulations reference a coordinate system that changes with deformation.

2. Are small angle approximations used? Few theories use exact kinematic relationships for highly nonlinear problems. When they do, other major simplifications are usually made so that the equations can be solved.

3. Is the mid-plane allowed to extend? If it is, what is the order of the mid-plane strain?

4. Does the theory allow for transverse shear deformation? If it does, what order is the representation and is locking a concern?

5. Are anisotropic materials allowed?

Huddleston [17] presents closed form solutions to various isotropic arches. His theory allows stretching of the mid-plane (to varying degrees), but does not include higher order strain terms. Strain is calculated from axial forces and moments through constitutive relations rather than strain displacement equations. Huddleston uses a total Lagrangian formulation with no small angle approximations. This theory is unique because Huddleston does not use finite elements to approximate solutions, rather, he solves the nonlinear partial differential equations simultaneously.

Mondkar and Powell [23] have developed a theory to predict the nonlinear dynamic and static response of general structures. Their method is quite advanced because they form the equations of motion in a total Lagrangian system and solve the equations by a numerical integration scheme while retaining many nonlinear terms. They utilize the principle of virtual displacement and apply a variational approach to the equations of motion to get an incremental form. After linearization and integration, the problem is discretized and placed into a classical finite element formulation. They do allow for mid-plane extensibility and transverse shear, but do not allow for anisotropic materials.

Hsiao and Hou [15] developed an updated Lagrangian formulation but are constrained by Euler-Bernolli beam assumptions. They don't include transverse shear, and assume constant membrane strain along the length. They are also constrained to small axial strain

(sum of membrane and bending strains). Hsiao and Hou use a corotational formulation to separate rigid body motion and deformations. This is a popular technique common to many updated Lagrangian systems. Their goal is to examine large rotations, but their beam assumptions make this a somewhat lower order theory (in strain).

Noguchi and Hisada [24] develop a nonlinear finite element model as an alternative to solving a linear eigenvalue problem to evaluate post collapse behavior of shell structures. They develop kinematic relationships through a vector oriented total Lagrangian approach. While the current effort isn't focused on buckling analysis, Noguchi and Hisada provide an excellent example of how large rotation kinematics can be derived for thin sections.

Karamanlidis, Honecker and Knothe [18] present a large deflection theory and finite element analysis for pre- and post-critical response of thin elastic frames. The thin section assumption allows them to neglect all transverse shear effects. They use an updated Lagrangian formulation with an attached reference frame that moves with deforming elements. Correction factors are applied to an energy function to account for theory limitations that would not ensure equilibrium. Using small increments, they are able to obtain solutions for very nonlinear configurations, but are limited to relatively thin structures.

Chandrashekara's [6] general approach to nonlinear bending of composite beams, plates and shells incorporates both geometric and material nonlinearity. Sanders shell theory provides the basis for his finite element evaluation. First order transverse shear is included, but Sanders elements are generally limited to lower order strain displacement relations. Shear correction factors are used to avoid locking. Chandrashekara considers geometric nonlinearity to the extent allowed by Von-Karman strain displacement equations. As a result, many higher order strain terms are neglected.

Sabir and Lock [31] examined geometric nonlinearity of circular arches made of isotropic material. They use a total Lagrangian formulation that neglects transverse shear strain. In-plane normal strain that is linear in the thickness coordinate is included with an extendible mid-plane, although many higher order mid-plane terms are neglected. They capture the multiple snapping phenomena common to transversely loaded thin arches.

DaDeppo and Schmidt [10], [11], [32] have conducted extensive research in the area of transversely loaded isotropic circular arches. In some problems they include mid-plane extensibility and transverse shear, while in others they do not. When included, transverse shear is constant through-the-thickness (no cross sectional warping), so shear correction factors are required to avoid locking in thin elements. Their theory is unique because they use an exact (no trigonometric approximations) total Lagrangian formulation with finite differencing to obtain equilibrium equations.

Epstein and Murray [12] neglect transverse shear strain, but retain up to quadratic order (in the thickness coordinate) of in-plane strain terms. They allow mid-plane extension and retain more higher order terms than most other theories. Epstein and Murray use a total Lagrangian formulation, but similar to DaDeppo and Schmidt, they use exact kinematic relationships. Their theory is simplified by considering only isotropic flat beams and keeping normals to the beam axis undistorted.

Belytschko and Glaum [4] use an updated Lagrangian in a corotational formulation. Coordinates are attached to elements during deformation allowing rigid body motion and deformation to be separated. Small angle approximations are used to relate the corotated axes. As the coordinate system for each element is updated at each increment, there is a limit on the size of the increment used. They utilize Euler-Bernoulli beam assumptions and neglect many of the higher order in-plane strain terms.

Brockman [5] reported on a three dimensional finite element code developed for the US Air Force. The program is capable of evaluating geometric as well as material

nonlinearity in general 3-D engineering structures. Brockman uses an updated Lagrangian reference frame to describe motion. He develops a 3-D solid element which is unique in that all transverse quantities are included. This advanced element does not include rotational degrees of freedom, but retains all the coupling between extensional and bending strains with complete mid-plane nonlinearity included. This theory is very complete and capable of very large displacements, but can be computationally intensive.

Antman [3] presents an advanced updated Lagrangian approach to nonlinear shell problems. He has geometrically exact kinematic relations, as no trigonometric approximations are ever made. This removes many of the traditional rotation limits that can exist from using trigonometric approximations. Antman includes mid-surface extension, transverse shear, bending rotation and transverse normal strain. Nonlinear partial differential equations are solved for several classes of problems. No finite element formulation is presented and solution techniques tend to be problem specific, making this theory difficult to extend to general structures.

Recently, Minguet and Dugundji [22] have developed a geometrically nonlinear model in an updated Lagrangian formulation that they validated with composite beam test data. They include constant transverse shear strain, mid-plane extensibility, and axial/shear coupling. Average rigid body motion is tracked by an attached reference frame via Euler angles. Minguet and Dugundji have conducted numerous tests on cantilevered composite beams in an effort to evaluate bending of composite helicopter blades. Their data provides an excellent opportunity to validate any nonlinear composite beam theory.

Creaghan [8] has developed a nonlinear composite arch model based on the shell theory of Palazotto and Dennis [27]. Creaghan includes mid-plane extensibility, parabolic transverse shear, and all nonlinear in-plane strain terms. His finite element model provides the initial background for the current effort.

In the specific area of large rotation, total Lagrangian theories seem to be using techniques traditionally used in updated Lagrangian systems. Nygard and Bergan [26] utilize a "ghost" reference system in a total Lagrangian formulation to more accurately capture large rotation effects. A corotational ghost reference system is used to capture rigid body motion between a deformed and the original coordinate systems. Strain is calculated in this rotated ghost frame, then transformed back to the original coordinate system (i.e., the total Lagrangian classification). This theory is accurate as long as rigid body motion alone separates the ghost and original coordinate systems. Nygard and Bergan do not use angle approximations, but introduce the complexity of multiple coordinate systems and additional strain transformations.

Problems are frequently classified by the size of the bending rotations. Nolte, Makowski and Stumpf [25] provide a classification theory to help determine when small, moderate, large and finite rotation theories should be used. According to [25], large rotation theory would be used when bending angles reached 15 degrees, and rotations near 50 degrees required a finite rotation theory with no approximations. Smith [33] applied their classifications to a typical cylindrical shell and showed that cross section rotations on the order of 5 degrees required the use of a moderate rotation theory. Furthermore, Smith showed that numerous authors used small or moderate rotation theory with seemingly good results when the Nolte criteria would require at least a large rotation theory. Few theories [3], [12] utilize finite rotations, but they usually require other limiting assumptions to formulate a tractable problem. Using a moderate rotation theory, Creaghan [8] showed accurate solutions up to 23 degrees of bending rotation. The goal of the current research is to obtain accurate solutions for bending rotations in excess of 45 degrees. In this area, more accurate rotation kinematics are necessary.

As can be seen from the discussion of the previous theories, nonlinear finite element models can become complicated. Simplifying assumptions help reduce complexity, but

nearly all modern efforts involve extensive calculations. With additional complexity usually comes additional computational requirements. With the advent of modern computers, theory complexity has increased as computing power has increased. Large systems coupled with complex programs can tax even the most modern serial machines. In order to examine possible benefits from recent vectorization and parallelization technologies, these subjects will be related to the current effort in the remainder of this chapter.

Recently, great progress has been made in applying parallel processing and vectorization to complex engineering problems. Adeli et al . [1] address high performance computing of structural mechanics problems. They examine vectorization techniques, programming language considerations, parallel numerical algorithms and performance parameters appropriate for evaluating improvements. Adeli et al. apply their concepts to numerous structural disciplines, such as linear and nonlinear structural analysis, transient analysis, dynamics of multi body flexible systems and structural optimization.

VanLuchene, Lee and Meyers [36] have evaluated the performance of large scale finite element solutions on vector processors. They define different levels of vectorization as they apply to different structural applications such as performance measurement, element calculations, system solution techniques and other related nonlinear topics. VanLuchene et al. boast a ten fold reduction in CPU time with proper vectorization techniques on typical nonlinear models.

Farhat, Wilson, and Powell [13] examine finite element codes on concurrent processing computers. They provide a parallel processing architecture specifically designed for finite element applications. Farhat et al. evaluate the entire finite element problem on parallel processors including domain subdivision, data structure and concurrent parallel solution algorithms. The current research will try to take advantage of some of these computational advances.

II. Theory

2.1 Kinematics

In order to examine the physical nature of the kinematic relationships, we will derive displacement equations in a vector format. While others present analytical functions that provide accurate results, they many times don't have physical meaning. The vector derivation provides the reader a better physical interpretation of the meaning of displacement terms and large rotation implications. We begin by deriving the displacement equations of a beam undergoing bending. Figure 2-1 shows a cross-section of a flat beam undergoing pure bending. Points j and k are on the outer surface with i being at the mid-plane. The \mathbf{V} coordinate system starts out parallel to the original system, but moves with the normal of the cross-section during deformation. Figure 2-1 shows the total Lagrangian coordinate system used with the $\hat{\mathbf{e}}$ coordinates as the original undeformed coordinate system (referred to as the global system). All displacements are measured relative to the global coordinate system. \mathbf{V}_{3i} is a vector directed along the cross-sectional thickness during bending with a magnitude equal to the thickness (h). The other two \mathbf{V} components form a right handed system with α being rotation angle about \mathbf{V}_{1i} . Since we are only currently considering bending (no warping from shear yet), \mathbf{V}_{3i} and \mathbf{V}_{2i} remain perpendicular throughout the deformation. Natural coordinates are utilized with ζ being aligned with the $\hat{\mathbf{e}}_3$ direction and having value +1 at the top surface and -1 at the bottom surface (as defined by positive coordinate direction).

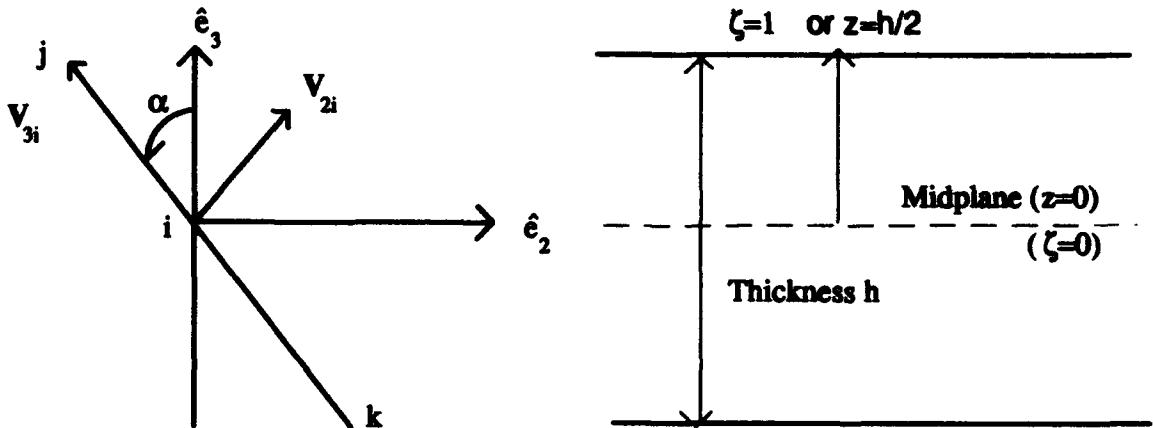


Figure 2-1 Flat Beam Bending Rotation

\mathbf{V}_{3i} can be expressed in global coordinates ($\hat{\mathbf{e}}_i$ frame) as:

$$\mathbf{V}_{3i} = (y_j - y_k) \hat{\mathbf{e}}_2 - (z_j - z_k) \hat{\mathbf{e}}_3 \quad (2-1)$$

The direction cosines for \mathbf{V}_{3i} are:

$$\begin{bmatrix} m_{3i} \\ n_{3i} \end{bmatrix} = \frac{1}{h} \begin{bmatrix} y_j - y_k \\ z_j - z_k \end{bmatrix} \quad (2-2)$$

So that Eqn (2-1) can be written as:

$$V_{3i}^n = h(m_{3i}^n \hat{\mathbf{e}}_2 + n_{3i}^n \hat{\mathbf{e}}_3) \quad (2-3)$$

Where the "n" refers to any deformed state. For any point P which lies along the cross-section, its position is a function of its location away from the beam center line and the location of point i on the mid-plane:

$$\begin{bmatrix} y \\ z \end{bmatrix}^n = \begin{bmatrix} y_i \\ z_i \end{bmatrix}^n + \frac{\zeta}{2} \mathbf{V}_{3i}^n \quad (2-4)$$

Now, for a total Lagrangian formulation, the displacement of any arbitrary point P can be expressed as:

$$\bar{\mathbf{u}}(z) = \begin{bmatrix} y_i \\ z_i \end{bmatrix}^n - \begin{bmatrix} y_i \\ z_i \end{bmatrix}^o + \frac{\zeta}{2} [\mathbf{V}_{3i}^n - \mathbf{V}_{3i}^o] \quad (2-5)$$

Where the "o" is the original undeformed state. We next assume that vertical displacement (w) does not vary but is constant through-the-thickness. The resulting motion of point P due to bending rotation is shown in figure 2-2.

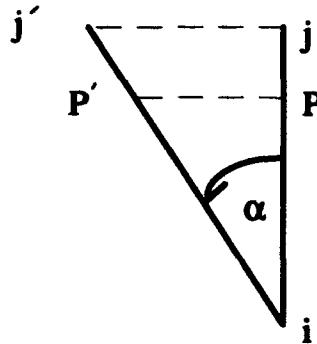


Figure 2-2 Bending Motion

The displacement assumptions shown in figure 2-2 would imply a lengthening of the beam normal. Any through-the-thickness normal strain from this is neglected. The deformation is taken as purely horizontal displacement to keep w constant through-the-thickness. The strain that develops does become significant at extremely large rotations (transverse normal strain becomes infinite as α approaches 90 degrees). Eqn(2-3) becomes:

$$V_{3i}^n = \frac{h}{\cos(\alpha_i)} \begin{bmatrix} m_{3i} \\ n_{3i} \end{bmatrix} \quad \text{where } m_{3i} = -\sin(\alpha_i) \text{ and } n_{3i} = \cos(\alpha_i) \quad (2-6)$$

Eqn(2-6) is substituted into Eqn(2-5) with $\alpha_o=0$ (the original system is aligned with itself):

$$\bar{u} = \bar{u}_i + \frac{\zeta}{2} h \left[\frac{1}{\cos(\alpha_i)} \begin{bmatrix} m_{3i} \\ n_{3i} \end{bmatrix}^n - \begin{bmatrix} m_{3i} \\ n_{3i} \end{bmatrix}^o \right]$$

so that (2-7)

$$V_{3i}^n = \begin{bmatrix} 1 & -\tan(\alpha_i) \\ 0 & 1 \end{bmatrix} V_{3i}^o = [R] V_{3i}^o$$

where \bar{u}_i is the motion of point P due to motion of mid-plane point i. $[R]$ is a bending rotation tensor. This derivation follows closely that of Noguchi and Hisada [24] who did a post buckling sensitivity analysis of shell structures. They use an updated Lagrangian with a truncated Taylor series expansion of the rotation tensor. A similar series expansion for the tangent in Eqn(2-7) will be shown later. Now, the general displacement equation can be expressed in natural coordinates as:

$$\bar{u} = \bar{u}_i + \frac{\zeta}{2} [[R] - [I]] V_{3i}^o \quad (2-8)$$

where $[I]$ is the 2X2 identity matrix.

The extension to curved beams, must account for the additional motion caused due to curvature. The \bar{u}_i term in Eqn (2-8) physically represents the displacement of an arbitrary point caused by motion of the mid-plane (non bending type of motion). The coordinates for the curved beam are shown in figure 2-3. The coordinates are an orthogonal curvilinear set with the 2 direction remaining along the arch mid-plane and the 3 direction pointing toward the center of curvature.

In a flat beam with Cartesian coordinates, the motion of the mid-plane gives a one to one correlation at point P, but in a curvilinear system, metric tensor coefficients must be applied [30], [27]. No longer can \bar{u}_i be represented directly by displacement of the mid-plane (v, w) only because there is a thickness dependence. This is shown physically and rather simply in figure 2-4.

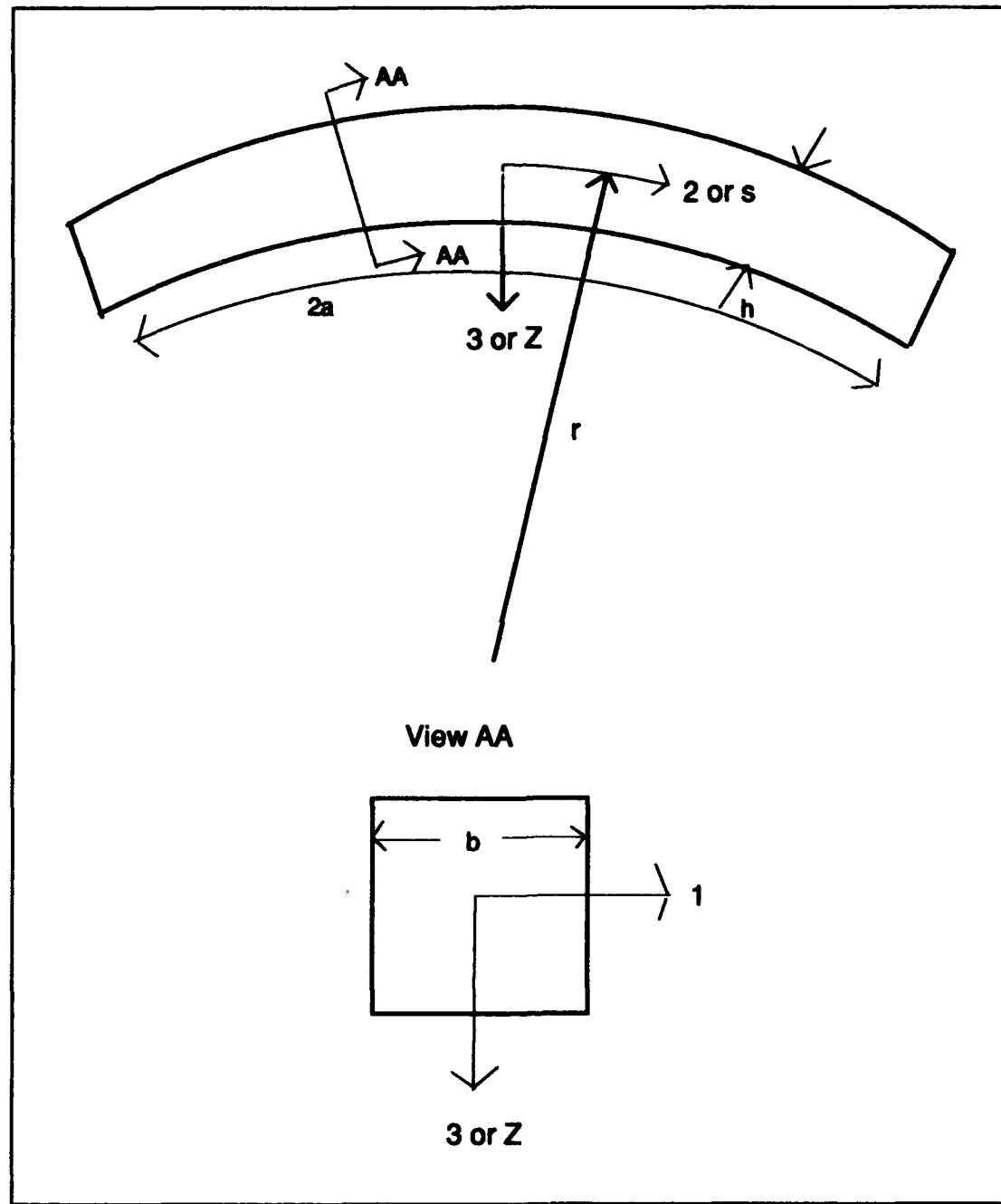


Figure 2-3 Global Coordinates of an Arch

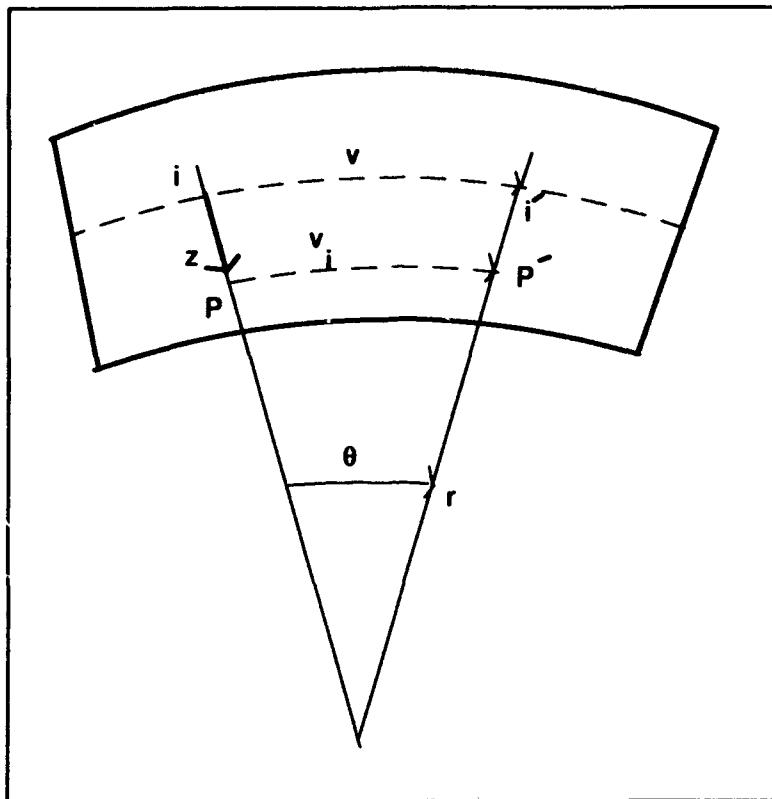


Figure 2-4 Mid-plane Displacement Effects

From simple geometric relations, it can be shown that the motion of point P due to motion of the mid-plane can be expressed as:

$$\bar{u}_i = \begin{bmatrix} v(1 - z/r) \\ w \end{bmatrix} \quad \text{where } v \text{ and } w \text{ are motion of the midplane point } i \quad (2-9)$$

The same result comes from proper application of the shell scale factors as shown in Saada [30] or Palazotto and Dennis [27].

We adopt the standard that a positive bending angle (ψ) is one that makes a point in the positive z direction move in a positive s direction. We defined α by using the right hand rule in a positive coordinate direction about the 1 axis which means $\psi = -\alpha$. For a graphical representation of the bending angle (ψ) and slope ($w_{,2}$) relationships, see figure 2-5.

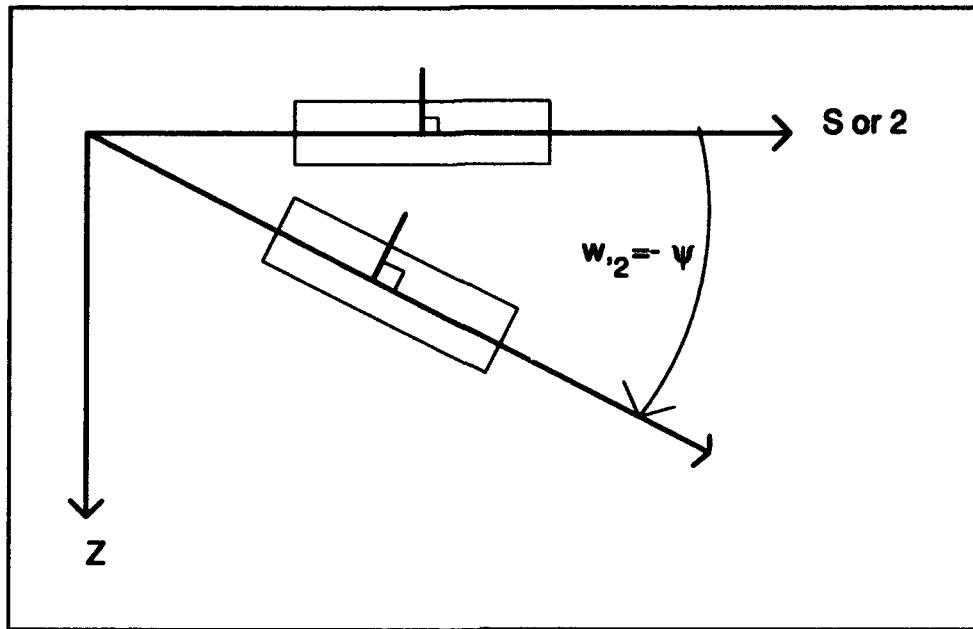


Figure 2-5 Pure Bending Convention

After Eqn(2-8) and Eqn(2-9) are expressed in global coordinates, the displacement components become:

$$u_2 = v \left(1 - \frac{z}{r}\right) + z \tan(\psi) \quad (2-10)$$

$$u_3 = w$$

Eqns(2-10) give the displacement relationships for bending movement only. It does not yet include any through-the-thickness shear deformation. Classical shell theory uses small angle approximations in the kinematics for the $\tan(\psi)$. Obviously, this theory is limited by extremely large bending rotations because in-plane displacements become infinite as ψ approaches 90 degrees. We would like to include higher orders of the z coordinate direction to capture displacement effects due to through-the-thickness shear strain. If we start with a common shell displacement relationship as found in Palazotto and Dennis [27], but include the tangent function in the z component instead of the small angle approximation ψ , then the displacements become:

$$u_1 = u = 0$$

$$u_2(s, z) = v \left(1 - \frac{z}{r} \right) + z \tan(\psi) + z^2 \phi + z^3 \gamma + z^4 \theta \quad (2-11)$$

$$u_3(s) = w$$

The determination of ϕ , γ and θ follows Palazotto and Dennis [27] with a few differences. The coefficients ϕ , γ and θ are determined by the boundary conditions of zero shear strain on the upper and lower surfaces. Strain will be discussed in section 2.3, but linear shear strain is used here to solve for the unknown constants in Eqn(2-11). Shear strain is found from applying linear Green strain and appropriate scale factors (details in section 2.3) and is evaluated at $z=\pm h/2$ resulting in:

$$\begin{aligned} \phi &= 0 & \theta &= \frac{\gamma}{2r} \\ \left[1 - \frac{h^2}{8r^2} \right] \gamma &= \frac{-4}{3h^2} (\tan(\psi) + w_{,2}) \approx \gamma \end{aligned} \quad (2-12)$$

Assuming that the beam is thin, i.e. $h/r \leq 1/5$, the $h^2/(8r^2)$ term can be neglected. γ is a coefficient representing the displacement due to transverse shear strain when multiplied by z^3 . To maintain consistency with linear shear strain, the small angle approximation ($\tan(\psi) \approx \psi$) is used for transverse shear resulting in:

$$\frac{-4}{3h^2} (\psi + w_{,2}) \approx \gamma \quad (2-13)$$

Typical arches considered have a large radius of curvature, making θ much smaller than γ , so it is neglected. Now the in-plane displacements (u_2) of Eqn 2-11 becomes:

$$u_2(s, z) = v \left(1 - \frac{z}{r} \right) + z \tan(\psi) + z^3 k (\psi + w_{,2}) \quad (2-14)$$

where $k = -4/(3h^2)$. We next define the shear angle as β . Figure 2-6 shows graphically the relationship between shear angle β and slope of the elastic curve $w_{,2}$. The β sign convention is the same as that for ψ , that is, a positive shear causes points on a positive z perpendicular to move in the positive s direction.

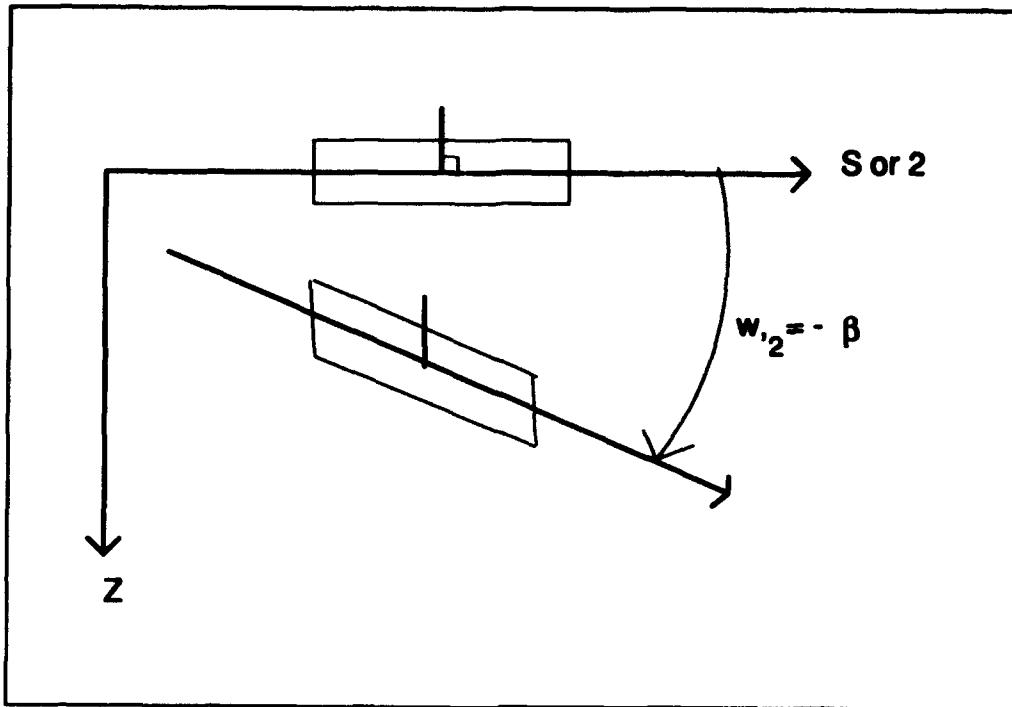


Figure 2-6 Pure Shear Convention

Mathematically, the three quantities w_2 , ψ , and β can be related by:

$$w_2 + \psi = -\beta \quad (2-15)$$

Figure 2-7 shows the sign convention of these three quantities together. The quantities β and ψ are treated as rotational angles about a point, while w_2 is the slope of a tangent to the elastic curve at the same point. In both cases in figure 2-7, w_2 is positive while β and ψ adhere to the sign conventions already described. Once w_2 and ψ are found from a finite element solution, the shear angle is calculated using Eqn(2-15).

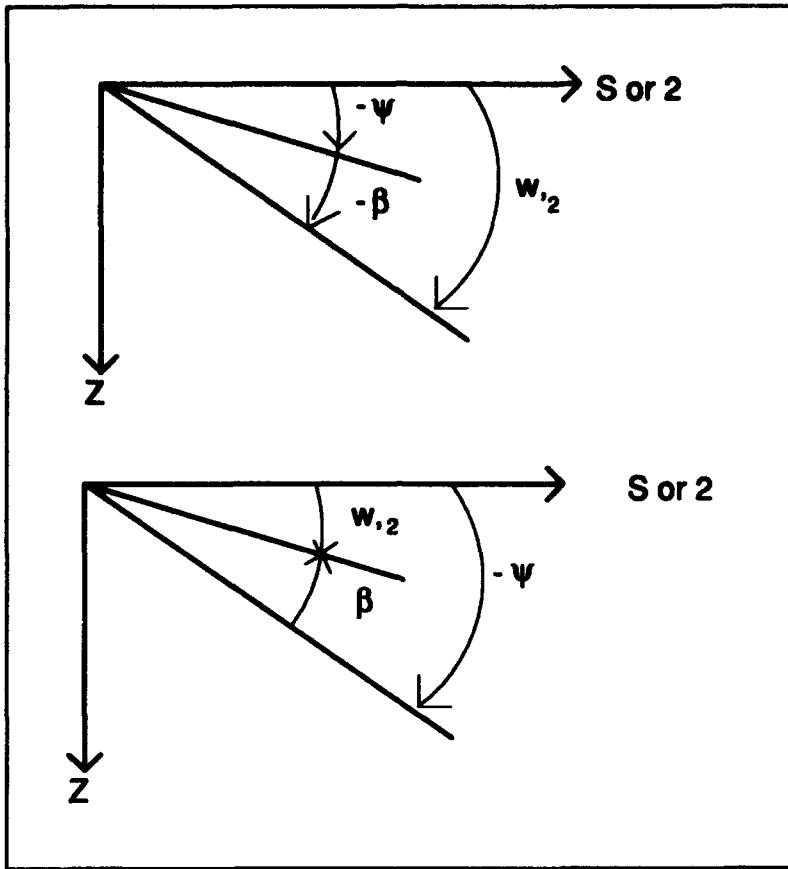


Figure 2-7 Slope, Shear and Bending Relationships

The tangent function in Eqn(2-14) is now approximated using a series expansion for bending. A series approximation is used to preserve the existing model degrees of freedom. The tangent of an angle can be expressed as [20]:

$$\tan(\psi) = \sum_{i=0}^{\infty} -1^{i-1} \frac{2^{2i} (2^{2i} - 1)}{2i!} B_{2i} \psi^{2i-1} \quad (2-16)$$

where the B_n 's are Bernoulli numbers ($B_1 = -1/2, B_2 = 1/6, B_3 = 0, B_4 = -1/30, B_5 = 0, B_6 = 1/42 \dots$) and ψ is expressed in radians. We are interested in bending rotations of approximately 50 degrees. At that angle, a two term truncation of Eqn(2-16) provides an approximation of $\tan(50^\circ)$ that is 92% accurate and is used here such that:

$$\tan(\psi) \approx \psi + \frac{1}{3}\psi^3 \quad (2-17)$$

Eqn(2-17) is substituted into Eqn(2-14) for u_2 with the other displacements unchanged leaving:

$$\begin{aligned} u_2(s, z) &= v \left(1 - \frac{z}{r} \right) + z \left(\psi + \frac{1}{3}\psi^3 \right) + z^3 k (\psi + w_{,2}) \\ u_3(s) &= w \\ u_1 &= 0 \end{aligned} \quad (2-18)$$

Where v and w are displacement of the mid-plane in the 2 and 3 global directions respectively, z is the coordinate distance away from the mid-plane, ψ is the angle due to cross-sectional bending rotation, and $w_{,2}$ is the slope of the mid-plane tangent. Eqns(2-18) are the final displacement relationships used for finding strain.

2.2 Constitutive Relations

Before beginning the stress strain relations used in the current work, it is necessary to define the contracted notation used for stress and strain components. Table 2.1 shows the explicit and contracted notation for both tensors.

Stress		Strain	
Contracted	Explicit	Contracted	Explicit
σ_1	σ_{11}	ϵ_1	ϵ_{11}
σ_2	σ_{22}	ϵ_2	ϵ_{22}
σ_3	σ_{33}	ϵ_3	ϵ_{33}
σ_4	σ_{23}	ϵ_4	$2\epsilon_{23}$
σ_5	σ_{13}	ϵ_5	$2\epsilon_{13}$
σ_6	σ_{12}	ϵ_6	$2\epsilon_{12}$

Table 2-1 Contraction Definitions

A composite lamina with embedded fibers has a local coordinate system with the 1' axis aligned with the fiber and the other two directions (2', 3') composing the transverse directions. The relationship between the material and global coordinate system is shown in figure 2-8. One should notice that the 3 direction points into the page and that 3 and 3' are always aligned for the present work.

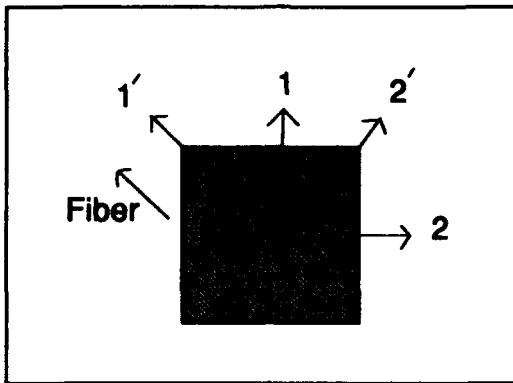


Figure 2-8 Material and Global Coordinate Relationships

The stress and strain for a single orthotropic lamina can be expressed in material coordinates as:

$$\begin{Bmatrix} \sigma'_1 \\ \sigma'_2 \\ \sigma'_3 \\ \sigma'_4 \\ \sigma'_5 \end{Bmatrix} = \begin{bmatrix} Q_{11} & Q_{12} & 0 & 0 & 0 \\ Q_{12} & Q_{22} & 0 & 0 & 0 \\ 0 & 0 & Q_{66} & 0 & 0 \\ 0 & 0 & 0 & Q_{44} & 0 \\ 0 & 0 & 0 & 0 & Q_{55} \end{bmatrix} \begin{Bmatrix} \epsilon'_1 \\ \epsilon'_2 \\ \epsilon'_3 \\ \epsilon'_4 \\ \epsilon'_5 \end{Bmatrix} \quad (2-19)$$

where σ'_i 's are ply stresses, ϵ'_i 's are strains and Q_{ij} 's are the plane stress reduced ply stiffness in the material coordinate system. The assumption for the present development follows shell theory and classical laminated plate theory closely. First, the beam is assumed thin, plane stress conditions approximate σ_3 as zero. The through thickness normal strain, ϵ_3 , is found through constitutive relationships as detailed in Palazotto and Dennis [27], but has already been incorporated Eqn (2-19). The Q_{ij} 's are found in Aagarwal et al. [2]:

$$\begin{aligned}
 Q_{11} &= \frac{E_1}{1-v_{12}v_{21}} & Q_{12} &= \frac{v_{12}E_2}{1-v_{12}v_{21}} = \frac{v_{21}E_1}{1-v_{12}v_{21}} \\
 Q_{22} &= \frac{E_2}{1-v_{12}v_{21}} & Q_{44} &= G_{23} \quad Q_{33} = G_{13} \quad Q_{66} = G_{12}
 \end{aligned} \tag{2-20}$$

where E_i 's are Young's moduli, G_{ij} 's are shear moduli, and v_{ij} 's are Poisson's ratios all expressed in material coordinates.

To represent Eqn(2-19) in global coordinates for a lamina, the Q_{ij} 's need to be transformed through the angle θ of figure 2-8. \bar{Q}_i 's represent plane stress ply stiffness represented in the global coordinate system. The resulting constitutive relationships for the k^{th} ply in global coordinates is:

$$\begin{Bmatrix} \sigma_1 \\ \sigma_2 \\ \sigma_6 \\ \sigma_4 \\ \sigma_5 \end{Bmatrix}_k = \begin{bmatrix} \bar{Q}_{11} & \bar{Q}_{12} & \bar{Q}_{16} & 0 & 0 \\ \bar{Q}_{12} & \bar{Q}_{22} & \bar{Q}_{26} & 0 & 0 \\ \bar{Q}_{16} & \bar{Q}_{26} & \bar{Q}_{66} & 0 & 0 \\ 0 & 0 & 0 & \bar{Q}_{44} & \bar{Q}_{46} \\ 0 & 0 & 0 & \bar{Q}_{64} & \bar{Q}_{55} \end{bmatrix}_k \begin{Bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_6 \\ \varepsilon_4 \\ \varepsilon_5 \end{Bmatrix}_k \tag{2-21}$$

where the expressions in the constitutive equations as functions of θ are found in appendix A.

Eqn (2-21) is simplified by first assuming the normal stress in the 1 direction (σ_1) is small due to the beam being narrow and is neglected. If this weren't true, the problem would better be modeled as a 2-D shell. We also assume the in-plane shear stress (σ_6) is small since the beam is narrow and traction free at the sides. Any coupling between normal and in-plane shear is not modeled here. Beam twisting is not included, therefore, σ_5 and ε_5 are neglected. If we solve the first three equations of Eqn(2-21) and eliminate ε_1 and ε_6 , the resulting simplified beam constitutive relations become:

$$\sigma_{44} = \bar{Q}_{44k} \varepsilon_{44}$$

$$\sigma_{24} = \hat{Q}_{24k} \varepsilon_{24}$$

where

(2-22)

$$\hat{Q}_{24} = \left(\frac{\bar{Q}_{12}^2 \bar{Q}_{44} - 2\bar{Q}_{16} \bar{Q}_{26} \bar{Q}_{11} + \bar{Q}_{26} \bar{Q}_{11}}{\bar{Q}_{16}^2 - \bar{Q}_{11} \bar{Q}_{44}} \right)_k$$

Since the model is a bending element, there is no consideration for twist or torsional stiffness. This limits the composite laminates to balanced symmetric lay-ups. This restriction makes \bar{Q}_{16} over the laminate zero. When each ply is added, the equations will simplify so that $(\bar{Q}_{16})_k$ can be set to zero in Eqn (2-22).

2.3 Strain Relations

Palazotto and Dennis [27] relate the Green strain and shell physical strains by:

$$\varepsilon_{ij} = \frac{\gamma_{ij}}{h_i h_j} \quad (\text{no sum}) \quad (2-23)$$

where γ_{ij} 's are components of the Green strain tensor and h_i 's are shell scale factors arising from the metric tensor for the coordinate system shown in figure 2-3. As discussed previously, only the in-plane normal strain and through-the-thickness shear strain are retained for the present work.

$$\begin{aligned} \gamma_{22} = & h_2 u_{2,2} + \frac{h_2 u_3}{h_3} h_{2,3} + \frac{h_2 u_1}{h_1} h_{2,1} \\ & + \frac{1}{2} \left(u_{2,2} + \frac{u_3}{h_3} h_{2,3} + \frac{u_1}{h_1} h_{2,1} \right)^2 \\ & + \frac{1}{2} \left(u_{3,2} - \frac{u_2}{h_3} h_{2,3} \right)^2 + \frac{1}{2} \left(u_{1,2} - \frac{u_2}{h_1} h_{2,1} \right)^2 \end{aligned} \quad (2-24)$$

and

$$\begin{aligned}
\gamma_{23} = & \frac{1}{2} \left(h_3 u_{3,2} + h_2 u_{2,3} - u_2 h_{2,3} - u_3 h_{3,2} \right) \\
& + \frac{1}{2} \left(u_{2,3} - \frac{u_3}{h_2} h_{2,3} \right) \left(u_{2,2} + \frac{u_3}{h_3} h_{2,3} + \frac{u_1}{h_1} h_{2,1} \right) \\
& + \frac{1}{2} \left(u_{3,2} - \frac{u_2}{h_3} h_{2,3} \right) \left(u_{3,3} + \frac{u_2}{h_2} h_{3,2} + \frac{u_1}{h_1} h_{3,1} \right) \\
& + \frac{1}{2} \left(u_{1,2} - \frac{u_2}{h_1} h_{2,1} \right) \left(u_{1,3} - \frac{u_2}{h_1} h_{3,1} \right)
\end{aligned} \tag{2-25}$$

where u_i are global displacements of Eqn (2-18).

For the arch problems considered, the scale factors are the same as those for a cylindrical shell as the beam width doesn't affect scale factors. For a cylinder with curvature in the 2 or s direction only, the scale factors are:

$$\begin{aligned}
h_1 &= h_3 = 1 \\
h_2 &= 1 - \frac{z}{r}
\end{aligned} \tag{2-26}$$

where the scale factors match those developed graphically in figure 2-4.

Now Eqn(2-26), Eqn(2-24) and Eqn(2-18) are placed in Eqn(2-23) such that:

$$\varepsilon_2 = \frac{\gamma_{23}}{h_2^2} \tag{2-27}$$

with all terms of Eqn(2-24) being retained since in-plane strain will dominate a thin beam bending problem. Palazotto and Dennis [27] neglect thirteen higher order terms. Smith [33] and Creaghan [8] each retained these terms, as does the present work (an extension of Creaghan's). They are retained here for completeness and for easy comparison to show the effects of the new kinematics over previous work. During the expansion of Eqn(2-27) by way of Eqn(2-24), many shape factor terms and their derivatives are present which complicates the expression from the perspective of representing the strain as the

summation of degrees of freedom times constants and times z to various powers. To simplify this, truncated Taylor series expansions of the scale factors are used, resulting in sixty expressions found in Palazotto and Dennis. For example, one of the simpler approximations is:

$$\frac{1}{h_2^2} = \frac{1}{\left(1 - \frac{z}{r}\right)\left(1 - \frac{z}{r}\right)} \approx 1 + \frac{2z}{r} \quad (2-28)$$

Terms of higher than first order in z are truncated. Smith [33] retained quadratic terms in the scale factor approximations with approximately 20% stiffness increase. Only a few of the sixty shell scale factor approximations are needed for the current work. The scale factor approximations should be implemented after full expansion of the strain equations. A different result comes from applying scale factor approximations before expansion of the strain equations. The scale factor approximations needed for the present work are:

$$\begin{array}{lll} \frac{1}{h_2} \approx 1 + zc & \frac{h_{2,3}}{h_2 h_3} \approx c - c^2 z & \frac{1}{h_2^2} \approx 1 + 2zc \\ \frac{h_{2,3}}{h_2^2 h_3} \approx -c - 2c^2 z & \frac{h_{2,3}^2}{h_2^2 h_3^2} \approx c^2 + 2zc^3 & \frac{h_{2,3}^2}{h_2^2} \approx c^2 + 2zc^3 \\ h_{2,3}^2 \approx c^2 & \frac{h_{2,3}^2}{h_2} \approx c^2 + zc^3 & \frac{h_{2,3}}{h_2^2} \approx -c - 2zc^2 \end{array} \quad (2-29)$$

where $c=1/r$.

Now the in-plane normal strain (ϵ_2) can be expressed in terms of mid-plane strain (ϵ_2^o) and functions independent of z (χ_{2i} 's) such that:

$$\epsilon_2 = \epsilon_2^o + \sum_{p=1}^7 z^p \chi_{2p} \quad (2-30)$$

where

$$\epsilon_2 = v_{,2} - wc + .5(v_{,2}^2 + w_{,2}^2 + v^2 c^2 + w^2 c^2) + vw_{,2}c - v_{,2}wc$$

$$\begin{aligned}\chi_{21} = & \psi_{,2} - wc^2 + w_{,2}^2 + w^2 c^3 - c^2(v_{,2}w - vw_{,2}) + v\psi c^2 + v_{,2}\psi_{,2}c - c(\psi_{,2}w - \psi w_{,2}) + \psi_{,2}\psi^2 + v_{,2}\psi_{,2}\psi^2 \\ & - wc\psi_{,2}\psi^2 + (1/3)(c^2\psi\psi^3 + cw_{,2}\psi^3)\end{aligned}$$

$$\begin{aligned}\chi_{22} = & \psi_{,2}c + .5(\psi_{,2}^2 + \psi^2 c^2) + v\psi c^3 - 2c^2(\psi_{,2}w - \psi w_{,2}) + \psi_{,2}v_{,2}c - 1.5(c^4 v^2 + c^2 v_{,2}^2) - c^2 v_{,2} \\ & + 2c^3(v_{,2}w - vw_{,2}) + c(\psi_{,2}\psi^2 + \psi_{,2}\psi^2 v_{,2}) + \psi_{,2}\psi^2 + .5w_{,2}^2\psi^4 - 2c^2 w\psi_{,2}\psi^2 + (2/3)c^2 w_{,2}\psi^3 +\end{aligned}$$

$$(1/3)c^3 v\psi^3 + (1/3)c^2\psi^4 + (1/18)c^2\psi^6$$

$$\begin{aligned}\chi_{23} = & k(w_{,22} + \psi_{,2}) + c\psi_{,2}^2 + \psi^2 c^3 + kv_{,2}(w_{,22} + \psi_{,2}) + vkc^2(w_{,2} + \psi) - wkc(w_{,22} + \psi_{,2}) + w_{,2}kc(w_{,2} + \psi) \\ & + c^5 v^2 + c^3 v_{,2}^2 - 2c^2(c^2 v\psi + v_{,2}\psi_{,2}) - 2c^2 v_{,2}\psi_{,2}^2 + 2c w_{,2}^2\psi^2 + c w_{,2}^2\psi^4 - (2/3)c^3 v^2(c\psi - \psi) + \\ & (1/9)c^3\psi^6\end{aligned}\quad (2-31)$$

$$\begin{aligned}\chi_{24} = & kc(w_{,22} + \psi_{,2}) + vkc^3(w_{,2} + \psi) + 2kc^2(-ww_{,22} - w\psi_{,2} + w_{,2}^2 + w_{,2}\psi) + k\psi_{,2}(w_{,22} + \psi_{,2}) + \\ & \psi kc^2(w_{,2} + \psi) + v_{,2}kc(w_{,22} + \psi_{,2}) + k\psi_{,2}\psi^2(1 + w_{,22}) + (1/3)c^2 k\psi^2(\psi + w_{,2})\end{aligned}$$

$$\begin{aligned}\chi_{25} = & 2kc[\psi_{,2}(w_{,22} + \psi_{,2}) + \psi c^2(w_{,2} + \psi) - cv_{,2}(w_{,22} + \psi_{,2}) - c^3 v(w_{,2} + \psi) + \psi_{,2}\psi^2 w_{,22}] + 2kc[w_{,2}^2\psi^2 + \\ & (1/3)c^2\psi^4 + (1/3)c^2\psi^3 w_{,2}]\end{aligned}$$

$$\chi_{26} = k^2/2[w_{,22}^2 + 2w_{,22}\psi_{,2} + \psi_{,2}^2 + c^2(w_{,2}^2 + 2w_{,2}\psi + \psi^2)]$$

$$\chi_{27} = k^2 c[(w_{,22} + \psi_{,2})^2 + c^2(w_{,2} + \psi)^2]$$

The bold and underlined terms are all new based on the tangent series expansion for u_2 .

The bold terms are retained in the present analysis while the underlined terms are neglected as higher order. The criteria for neglecting the underlined terms was any combination of c , ψ , $\psi_{,2}$, and $w_{,2}$ that is order 5 or higher.

If the beam is thin, then in-plane stress and strain dominate the bending problem. Therefore, for simplicity, only the linear components of Eqn (2-25) are retained for

through-the-thickness shear strain. The strain ϵ_4 can be found by combining Eqns (2-23), (2-25) and (2-26):

$$\epsilon_4 = 2\epsilon_{23} = \frac{1}{1 - \frac{z}{r}} \left[u_{3,2} + \left(1 - \frac{z}{r} \right) u_{2,3} + \frac{u_2}{r} \right] \quad (2-32)$$

Since the shear strain is linear, it is appropriate to use small angle approximations for the global displacement functions in Eqn(2-32). Eqn(2-18) then becomes:

$$\begin{aligned} u_2(s, z) &= v \left(1 - \frac{z}{r} \right) + z(\psi) + z^3 k(\psi + w_{,2}) \\ u_3(s) &= w \\ u_1 &= 0 \end{aligned} \quad (2-33)$$

Placing Eqn(2-33) into Eqn(2-32) gives:

$$\epsilon_4 = \frac{1}{1 - \frac{z}{r}} \left[w_{,2} + \psi - \frac{4z^2}{h^2}(\psi + w_{,2}) + \frac{8z^3}{3h^2r}(\psi + w_{,2}) \right] \quad (2-34)$$

The largest z can get is $h/2$ making the term $8z^3/(3h^2r)$ comparatively small since $h/r \leq 1/5$ (1/15 the next smallest term) and is neglected. The scale factor term in the front of Eqn(2-34) is approximated by a truncated binomial series expansion. It is truncated for the same reason, namely, that the second term is at maximum 1/10 the first:

$$\frac{1}{1 - \frac{z}{r}} = 1 + \frac{z}{r} + \frac{z^2}{r^2 \left(1 - \frac{z}{r} \right)} + \dots \approx 1 \quad (2-35)$$

Now the through thickness shear strain can be expressed as:

$$\epsilon_4 = \epsilon_4^o + z^2 \chi_{,2}$$

with

$$\epsilon_4^o = w_{,2} + \psi \quad (2-36)$$

and

$$\chi_{,2} = \frac{-4}{h^2}(\psi + w_{,2})$$

where ϵ_{ij} is the mid-plane shear strain. Eqns(2-31) and Eqns(2-36) were generated by hand and then checked using the symbolic manipulator *Macsyma* [34].

2.4 Beam Potential Energy

The internal strain energy density of an elastic conservative system (reversible adiabatic or isothermal loading) with small strains is:

$$\frac{\partial W^*}{\partial \epsilon_{ij}} = \sigma_{ij} \quad (2-37)$$

where W^* is strain energy density, ϵ_{ij} the strain tensor, and σ_{ij} is the stress tensor. Stress and strain can be related by a constitutive law with an elasticity tensor (a_{ijkl}):

$$\sigma_{ij} = a_{ijkl} \epsilon_{kl} \quad (2-38)$$

The strain energy density can now be defined by:

$$W^* = \frac{1}{2} a_{ijkl} \epsilon_{ij} \epsilon_{kl} \quad (2-39)$$

The potential energy Π_p is the sum of the internal strain energy U and the work done by external forces V :

$$\Pi_p = U + V \quad (2-40)$$

where U is defined by integrating the strain energy density over the volume. Since the current problem retains two strain components, the beam strain energy can be divided into contributions from in-plane strain U_1 and through-the-thickness shear strain U_2 :

$$U = \int_{Vol} W^* dV = U_1 + U_2 \quad (2-41)$$

where for a beam width b and length l , U_1 and U_2 are defined as:

$$U_1 = \frac{1}{2} b \int_{-h/2}^{h/2} \hat{Q}_{2k} \varepsilon_2^2 dz ds \quad (2-42)$$

$$U_2 = \frac{1}{2} b \int_{-h/2}^{h/2} \bar{Q}_{4k} \varepsilon_4^2 dz ds \quad (2-43)$$

From the representation of ε_2 in Eqn(2-30) we have:

$$\varepsilon_2^2 = (\varepsilon_2^0)^2 + 2 \sum_{p=1}^7 \varepsilon_2^p \chi_{2p} z^p + \sum_{p=1}^7 (\chi_{2p} z^p)^2 \quad (2-44)$$

So that U_1 can be divided into three parts corresponding to the three strain components of Eqn(2-44):

$$U_1 = \frac{1}{2} b \int_{-h/2}^{h/2} (\mu_1 + \mu_2 + \mu_3) ds \quad (2-45)$$

where:

$$\begin{aligned} \mu_1 &= \int_{-h/2}^{h/2} \hat{Q}_{2k} (\varepsilon_2^0)^2 dz = A (\varepsilon_2^0)^2 \\ \mu_2 &= \int_{-h/2}^{h/2} 2 \hat{Q}_{2k} (\varepsilon_2^0) \sum_{p=1}^7 \chi_{2p} z^p dz = 2 \varepsilon_2^0 (B \chi_{21} + D \chi_{22} + E \chi_{23} + F \chi_{24} + G \chi_{25} + H \chi_{26} + I \chi_{27}) \\ \mu_3 &= \int_{-h/2}^{h/2} \hat{Q}_{2k} \left(\sum_{p=1}^7 \chi_{2p} z^p \right)^2 dz = D \chi_{21} \chi_{21} + 2E \chi_{21} \chi_{22} + F (2 \chi_{21} \chi_{23} + \chi_{22} \chi_{22}) \\ &\quad + 2G (\chi_{21} \chi_{24} + \chi_{22} \chi_{23}) \\ &\quad + 2H (\chi_{21} \chi_{25} + \chi_{22} \chi_{24}) + 2I (\chi_{21} \chi_{26} + \chi_{22} \chi_{25} + \chi_{23} \chi_{24}) \\ &\quad + J (2 \chi_{21} \chi_{27} + 2 \chi_{22} \chi_{26} + 2 \chi_{23} \chi_{25} + \chi_{24}^2) + 2K (\chi_{22} \chi_{27} + \chi_{23} \chi_{26} + \chi_{24} \chi_{25}) \\ &\quad + L (2 \chi_{23} \chi_{27} + 2 \chi_{24} \chi_{26} + \chi_{25}^2) + 2P (\chi_{24} \chi_{27} + \chi_{25} \chi_{26}) + R (2 \chi_{25} \chi_{27} + \chi_{26}^2) \\ &\quad + 2S \chi_{26} \chi_{27} + T \chi_{27}^2 \end{aligned} \quad (2-46)$$

The strain energy components of Eqns(2-46) have been integrated through-the-thickness to obtain the elasticity terms ($A, B, D, E, F, G, H, I, J, K, L, P, R, S, T$) which are defined by:

$$(A, B, D, E, F, G, H, I, J, K, L, P, R, S, T) = \int_{-h/2}^{h/2} \hat{Q}_{2k} (1, z, z^2, z^3, z^4, z^5, z^6, z^7, z^8, z^9, z^{10}, z^{11}, z^{12}, z^{13}, z^{14}) dz \quad (2-47)$$

All other terms are independent of the thickness direction. For a composite, \hat{Q}_{2k} will vary through-the-thickness (can be different at any k^{th} ply). As already discussed, the present method is limited to balanced symmetric lay ups which means the integration of any odd powers of z in Eqn(2-47) will be zero. This makes the elasticity terms (B, E, G, I, K, P, S) zero when integrated through the beam thickness which simplifies Eqn(2-46).

The strain energy due to through-the-thickness shear strain can be represented in a similar manner. Substitution of Eqn(2-36) into Eqn(2-43) gives:

$$U_2 = \frac{1}{2} b \int_I \mu_s ds \quad (2-48)$$

where,

$$\begin{aligned} \mu_s &= \int_{-h/2}^{h/2} \bar{Q}_{44k} \left[(\varepsilon_4^o)^2 + 2\varepsilon_4^o \chi_{42} z^2 + (\chi_{42})^2 z^4 \right] dz \\ &= AS (\varepsilon_4^o)^2 + 2DS \varepsilon_4^o \chi_{42} + FS (\chi_{42})^2 \end{aligned} \quad (2-49)$$

The shear elasticity terms (AS, DS, FS) are found in a manner similar to Eqn(2-47) by integrating through-the-thickness:

$$[AS, DS, FS] = \int_{-h/2}^{h/2} \bar{Q}_{44k} [1, z^2, z^4] dz \quad (2-50)$$

Next we represent the strain expressions of Eqn(2-30) and Eqn(2-36) in a more convenient form to place into the strain energy representations of Eqn(2-46) and Eqn(2-49). Strain can be divided into linear and nonlinear parts by first defining the gradient

vector d . The displacement gradient vector contains variables necessary to represent the strain expressions of Eqn(2-30) and Eqn(2-36):

$$d^T = \{v \ v_{,1} \ w \ w_{,1} \ w_{,22} \ \psi \ \psi_{,1}\} \quad (2-51)$$

The terms in d are exactly the ones needed for strain displacement equations. The in-plane strain can be represented as:

$$\begin{aligned} \epsilon_2^o &= L_o^T d + \frac{1}{2} d^T H_o d \\ \chi_{2,p} &= L_p^T d + \frac{1}{2} d^T H_p d \end{aligned} \quad (2-52)$$

where $p=1,2,3,4,5,6,7$. The L_i 's are column vectors of constants to represent the linear terms of strain and the H_i 's are symmetric matrices that represent the nonlinear strain components when combined with the displacement gradient vector as shown above.

The strain ϵ_4 can be represented in a similar manner. The shear expressions are much simpler since only linear strain is used and small angle approximations were used in the kinematics equations.

$$\begin{aligned} \epsilon_4^o &= S_o^T d \\ \chi_{42} &= S_2^T d \end{aligned} \quad (2-53)$$

where the S_i 's are column vectors of constants found in appendix A.

Creaghan had constants in all of his H and L matrices where the current formulation contains some nonlinear terms of ψ . The use of this representation is demonstrated below with the complete representation found in appendix A.

$$\varepsilon_2^o = \{0 \ 1 \ -c \ 0 \ 0 \ 0 \ 0\} \begin{Bmatrix} v \\ v_{,2} \\ w \\ w_{,2} \\ w_{,22} \\ \psi \\ \psi_{,2} \end{Bmatrix} + \quad (2-54)$$

$$+ \frac{1}{2} \{v \ v_{,2} \ w \ w_{,2} \ w_{,22} \ \psi \ \psi_{,2}\} \begin{Bmatrix} c^2 & 0 & 0 & c & 0 & 0 & 0 \\ 0 & 1 & -c & 0 & 0 & 0 & 0 \\ 0 & -c & c^2 & 0 & 0 & 0 & 0 \\ c & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{Bmatrix} \begin{Bmatrix} v \\ v_{,2} \\ w \\ w_{,2} \\ w_{,22} \\ \psi \\ \psi_{,2} \end{Bmatrix}$$

As mentioned above, many of the H_i 's are no longer matrices of constants but have become nonlinear in ψ due to inclusion of large rotation kinematics. For example, χ_{21} is represented as:

$$\chi_{21} = \{0 \ 0 \ -c^2 \ 0 \ 0 \ 0 \ 1\} \begin{Bmatrix} v \\ v_{,2} \\ w \\ w_{,2} \\ w_{,22} \\ \psi \\ \psi_{,2} \end{Bmatrix} + \quad (2-55)$$

$$\frac{1}{2} \{v \ v_{,2} \ w \ w_{,2} \ w_{,22} \ \psi \ \psi_{,2}\} \begin{Bmatrix} 0 & 0 & 0 & c^2 & 0 & c^2 & 0 \\ 0 & 0 & -c^2 & 0 & 0 & 0 & 1+\psi^2 \\ 0 & -c^2 & 2c^3 & 0 & 0 & 0 & -c(1+\psi^2) \\ c^2 & 0 & 0 & 2c & 0 & c & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c^2 & 0 & 0 & c & 0 & 0 & \psi \\ 0 & 1+\psi^2 & -c(1+\psi^2) & 0 & 0 & \psi & 0 \end{Bmatrix} \begin{Bmatrix} v \\ v_{,2} \\ w \\ w_{,2} \\ w_{,22} \\ \psi \\ \psi_{,2} \end{Bmatrix}$$

One should notice that H_1 is now composed of constants and terms involving ψ . The H_i 's are still treated as though they are constant matrices instead of higher order functions of the displacement gradient vector. This is done to keep the form of the problem consistent with Palazotto and Dennis [27] and Creaghan [8]. The H_i 's are linearized in ψ by substituting in the last incremental values of ψ . Other entries in the displacement gradient vector could have been represented in the H_i 's (since all the new terms are at least order 3 of the displacement gradient vector terms), but ψ was used to keep the formulation simple with only one variable making the matrices nonlinear. This will make taking the first variation of the potential energy somewhat simpler later.

The strain representations in Eqns(2-52) and (2-53) can be placed into the strain energy representations of Eqns(2-49), (2-48), (2-45), (2-46) and finally Eqn(2-41), so that the total beam strain energy can be expressed in terms of the displacement gradient vector as:

$$U = \frac{1}{2} b \int_l d^T \left[\hat{K} + \frac{\hat{N}_1(d)}{3} + \frac{\hat{N}_2(d)}{6} \right] d \, ds \quad (2-56)$$

where \hat{K} is a matrix of constant terms, \hat{N}_1 is a matrix composed of linear functions of d , and \hat{N}_2 contains quadratic terms from d . The form of Eqn(2-56) is desired for the ease of finite element formulation based on minimizing the potential energy function to come. Again, there are higher order functions of ψ in \hat{N}_1 and \hat{N}_2 , but are treated as constants. Eqn(2-56) is put in the shown form by making the aforementioned substitutions then taking advantage of symmetry. This process is shown in Palazotto and Dennis [27:74] and is followed exactly. The matrices \hat{K} , \hat{N}_1 , and \hat{N}_2 are represented in appendix A in terms

of L_i 's, H_i 's and elasticity terms. Since forming these stiffness arrays involves literally thousands of operations, *Macsyma* [34] was used to do the manipulations and then generate the FORTRAN. The *Macsyma* code used to do these operations is located in appendix B.

So far, the present formulation has been independent of a solution method. The present problem is a set of nonlinear ordinary differential equations that would be formed by setting the first variation of the potential energy to zero. Instead of solving a system of nonlinear ODE's, the system is discretized into elements connected by nodes. The result of the beam discretization is a system of nonlinear algebraic equations which can be linearized and solved iteratively giving approximate problem solutions.

2.5 Finite Element Formulation

The nonlinear algebraic equations that result by casting the problem in a finite element form can be solved in a linear fashion by using an incremental approach to achieving equilibrium. Two methods are presently used: one varies the external load and the other varies the displacement. Each is discussed in section 2-6. The beam finite element developed for the current formulation is shown in figure 2-9. This element is the same as used by Creaghan [8].

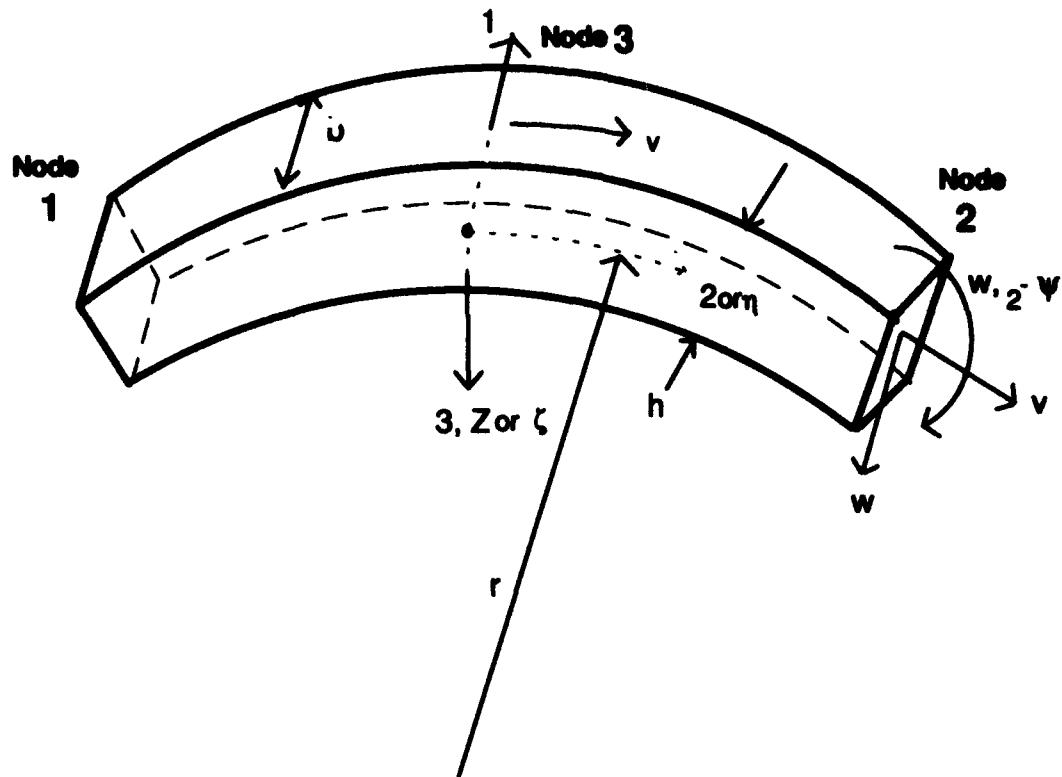


Figure 2-9 Beam Finite Element

The element in figure 2-9 has three nodes with degrees of freedom (DOF) of v , w , ψ , and w_2 at the ends and v only at the middle node. The degrees of freedom (DOF) match quantities needed in the kinematic Eqns(2-18). The middle node v is included to help capture the axial extension and contraction to a higher order. Creaghan showed stiff solutions when this DOF was omitted, so it is retained here. Only w has derivatives in the DOF. In order to achieve w_2 continuity between the elements, Hermitian shape functions are used for w and its derivatives achieving C^1 continuity[7]. The other DOF don't have any derivatives as DOF meaning only C^0 continuity is required. Linear shape functions are used to interpolate ψ , and quadratic shape functions are used to interpolate v . The values of the DOF $q(\eta)$ at any point in the beam can be expressed in natural coordinates in terms of the nodal DOF q by:

$$q(\eta) = \begin{bmatrix} v \\ \psi \\ w \\ w_{,2} \end{bmatrix} = \begin{bmatrix} Q_1 & 0 & 0 & 0 & Q_3 & Q_2 & 0 & 0 & 0 \\ 0 & N_1 & 0 & 0 & 0 & 0 & N_2 & 0 & 0 \\ 0 & 0 & H_{11} & H_{12} & 0 & 0 & 0 & H_{21} & H_{22} \\ 0 & 0 & H_{11,\eta} & H_{12,\eta} & 0 & 0 & 0 & H_{21,\eta} & H_{22,\eta} \end{bmatrix} \begin{bmatrix} v^{(1)} \\ \psi^{(1)} \\ w^{(1)} \\ w_{,2}^{(1)} \\ v^{(3)} \\ v^{(2)} \\ \psi^{(2)} \\ w^{(2)} \\ w_{,2}^{(2)} \end{bmatrix} \quad (2-57)$$

where the shape functions of Eqn(2-57) are:

$$\begin{aligned} Q_1 &= \frac{1}{2}(\eta^2 - \eta) & Q_2 &= \frac{1}{2}(\eta^2 + \eta) & Q_3 &= (1 - \eta^2) \\ N_1 &= \frac{1}{2}(1 - \eta) & N_2 &= \frac{1}{2}(1 + \eta) \\ H_{11} &= \frac{1}{4}(2 - 3\eta + \eta^3) & H_{12} &= \frac{a}{4}(1 - \eta - \eta^2 + \eta^3) \\ H_{21} &= \frac{1}{4}(2 + 3\eta - \eta^3) & H_{22} &= \frac{a}{4}(-1 - \eta + \eta^2 + \eta^3) \end{aligned} \quad (2-58)$$

To use the potential energy expression of Eqn(2-56), we must relate the displacement gradient vector, Eqn(2-51), to the nodal degrees of freedom for the element. One should notice that ψ is interpolated linearly. It will be shown in chapter III that this element has some solution problems and breaks down for deep arches. One way to help alleviate this problem is to use a higher order interpolation of the bending angle. This could be accomplished in a similar manner as w with C^1 functions, or add more ψ DOF's to the element. An initial effort to incorporate large bending rotation involved "scaling" the shape functions that interpolated ψ to a value near the tangent. In this case, locking occurred because transverse shear was updated also. The matrix of shape functions in natural coordinates used to get d will be referred to as D .

$$d(\eta) = Dq = \begin{bmatrix} Q_1 & 0 & 0 & 0 & Q_3 & Q_2 & 0 & 0 & 0 \\ Q_{1,\eta} & 0 & 0 & 0 & Q_{3,\eta} & Q_{2,\eta} & 0 & 0 & 0 \\ 0 & 0 & H_{11} & H_{12} & 0 & 0 & 0 & H_{21} & H_{22} \\ 0 & 0 & H_{11,\eta} & H_{12,\eta} & 0 & 0 & 0 & H_{21,\eta} & H_{22,\eta} \\ 0 & 0 & H_{11,\eta,\eta} & H_{12,\eta,\eta} & 0 & 0 & 0 & H_{21,\eta,\eta} & H_{22,\eta,\eta} \\ 0 & N_1 & 0 & 0 & 0 & 0 & N_2 & 0 & 0 \\ 0 & N_{1,\eta} & 0 & 0 & 0 & 0 & N_{2,\eta} & 0 & 0 \end{bmatrix} \begin{bmatrix} v^{(1)} \\ \psi^{(1)} \\ w^{(1)} \\ w_{\eta}^{(1)} \\ v^{(3)} \\ v^{(2)} \\ \psi^{(2)} \\ w^{(3)} \\ w_{\eta}^{(2)} \end{bmatrix} \quad (2-59)$$

Eqn(2-59) is placed into global coordinates by multiplying by the inverse of the Jacobian matrix (J).

$$d(s) = J^{-1}d(\eta) = J^{-1}Dq = Tq \quad (2-60)$$

where J^{-1} is defined by:

$$J^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{a} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{a}{a} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{a} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{a^2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{a} \end{bmatrix} \quad (2-61)$$

where a is half the element length.

The strain energy of the beam from Eqn(2-56) can be expressed in terms of the nodal degrees of freedom by:

$$U = \frac{1}{2} b \int_I q^T T^T \left[\hat{K} + \frac{\hat{N}_1}{3} + \frac{\hat{N}_2}{6} \right] T q \, ds \quad (2-62)$$

We can define new quantities which will make up a tangent stiffness matrix by integrating along the beam length such that:

$$K = b \int_I T^T \hat{K} T \, ds \quad N_1 = b \int_I T^T \hat{N}_1 T \, ds \quad N_2 = b \int_I T^T \hat{N}_2 T \, ds \quad (2-63)$$

By substituting Eqns(2-63) into Eqn(2-62) and recalling from Eqn(2-40) that the beam potential energy is the sum of internal strain energy and the work of external forces, the potential energy for a finite element can be expressed as:

$$\Pi_p = \frac{1}{2} q^T \left[K + \frac{N_1(q)}{3} + \frac{N_2(q)}{6} \right] q - q^T R \quad (2-64)$$

where R is a vector of externally applied forces.

Carrying out the first variation of Eqn(2-64) and setting it to zero results in equilibrium equations (F) below.

$$\delta \Pi_p = \delta q^T \left[\left[K + \frac{N_1(q)}{2} + \frac{N_2(q)}{3} \right] q - R \right] = \delta q^T [F(q)] = 0 \quad (2-65)$$

For an arbitrary and independent δq , $F(q) = 0$ is a set of nonlinear algebraic equilibrium equations for the nodal degrees of freedom. Eqn(2-65) is solved by using linearized iterative techniques. These types of solutions are derived by expanding Eqn(2-65) in a truncated Taylor series for a small Δq such that:

$$F(q + \Delta q) \approx F(q) + \frac{\partial F}{\partial q} \Delta q = 0 \quad (2-66)$$

or:

$$\frac{\partial F}{\partial q} \Delta q = -F(q) \quad (2-67)$$

Now, by taking the expression for F from Eqn(2-65) and placing it with appropriate derivatives into Eqn(2-67), one obtains:

$$K_T \Delta q = - \left[K + \frac{N_1}{2} + \frac{N_2}{3} \right] q + R \quad (2-68)$$

where

$$K_T = [K + N_1 + N_2]$$

Eqn(2-68) is the final equation desired in order to attain the values of the equilibrium degrees of freedom. Whatever solution method is used, the stiffness arrays are geometrically nonlinear and change with load/displacement of the continuos structure.

2.6 Numerical Solution Methods

Eqn(2-68) can be rewritten to include all the elements of a particular model by:

$$\sum_{j=1}^n \left[b \int_I T^T (\hat{K} + \hat{N}_1 + \hat{N}_2) T ds \right]_j \Delta q = - \sum_{j=1}^n \left[b \int_I T^T \left(\hat{K} + \frac{\hat{N}_1}{2} + \frac{\hat{N}_2}{3} \right) T ds \right]_j q + R \quad (2-69)$$

where n is the number of elements and j represents the integration over a specific element. The above equations are placed into a natural coordinate system by changing the limits of integration to ± 1 and including the determinant of the Jacobian to facilitate the use of Gauss quadrature numerical integration. The current model has the capability to handle Gauss quadrature up to order 7, but order 5 was used for all the problems discussed here. Order m Gauss quadrature is required to integrate polynomials of order $2m-1$. T contains cubic polynomials and \hat{N}_2 is quadratic in displacement containing sixth order polynomials. To integrate exactly would require Gauss quadrature of order 7. Cook [7] shows little difference in solutions between using 5 and 7 point Gauss quadrature.

Figure 2-10 shows a generic load displacement curve with limit points. As the slope of the curve approaches 0 (point A), the structure collapses or snaps. Beyond this point, load decreases even though displacement continues to increase. As we continue down the equilibrium path, a vertical tangent is passed (point B) where the tangent stiffness matrix becomes singular. After the structure has "snapped," at some point under an inverted configuration, the structure will start to carry increasing load again. This transition happens at another horizontal tangent (point C).

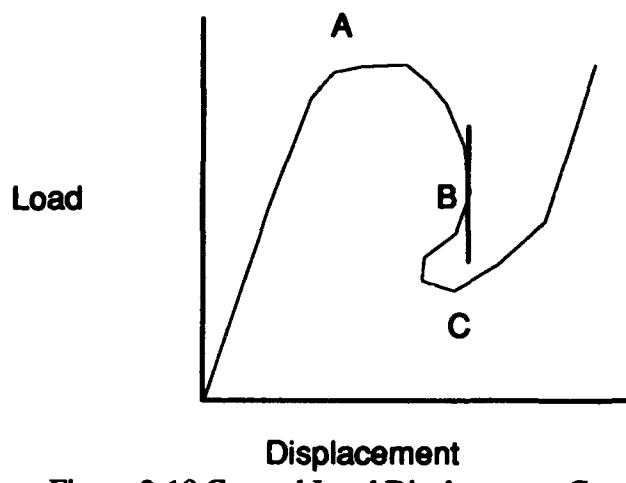


Figure 2-10 General Load Displacement Curve

For the current type of nonlinear problems considered, Newton-Ralphson approaches are used. Two techniques that were developed elsewhere are used here. Each is explained, but for further details, the reader should refer to the original development. The first solution scheme is a displacement control method developed by Palazotto and Dennis [27]. In displacement control method, displacement is incremented and load iterated until equilibrium is achieved. Displacement control is good for traversing limit load points like point A in figure 2-10, but cannot handle points like B since there is not a unique solution to a single prescribed displacement. Displacement control would give solutions that traverse instantaneously from B to C and do not capture the snap back characteristic shown in figure 2-10. Many times, points with vertical tangents can be traversed by

making large enough displacement increments to "jump" past the problem area. Palazotto and Dennis [27] develop the displacement control algorithm as:

$$\begin{aligned} & [K + N_1(q_{n-1}) + N_2(q_{n-1}^2)]\Delta q_{n-1} = \\ & - \left[K + \frac{N_1(q_{n-1})}{2} + \frac{N_2(q_{n-1}^2)}{3} \right] q_{n-1} + R_1 \end{aligned} \quad (2-70)$$

where n is the iteration number. Eqn(2-70) is for a single displacement increment. Figure 2-11 shows graphically how displacement control works. Beginning from an established equilibrium point, the displacement is incremented, and load found from Eqn(2-70). A new displacement value is then found by satisfying equilibrium with a given tolerance. Palazotto and Dennis compare a displacement out of balance condition to a user supplied criteria.

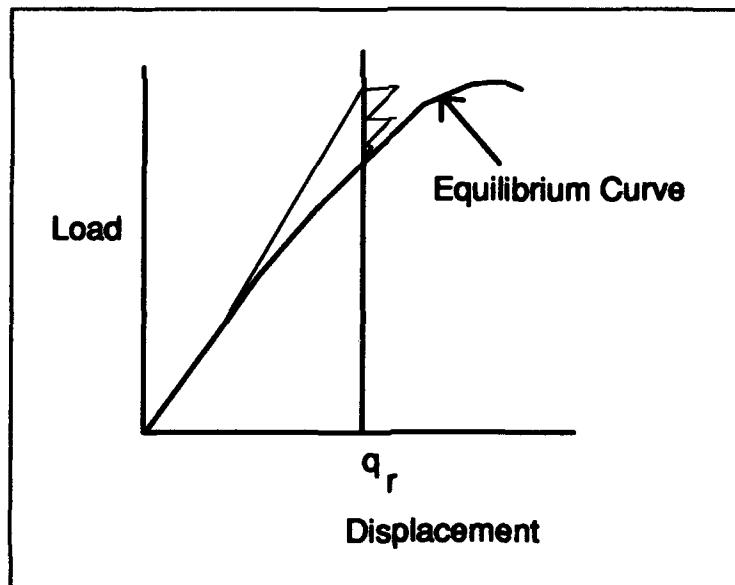


Figure 2-11 Displacement Control

The second Newton-Ralphson solution method is a modified Riks-Wempner technique which was adapted for the current problem by Tsai and Palazotto [35] from an original

procedure shown by Crisfield [9]. The Riks method, shown below, will capture multiple snap through points of an equilibrium curve. It does this by introducing a loading parameter, λ , such that the equilibrium equation becomes:

$$F(q, \lambda) = \left(K + \frac{N_1}{2} + \frac{N_2}{3} \right) q - \lambda R = 0 \quad (2-71)$$

Eqn(2-71) is expanded in a manner similar to the equilibrium equation for displacement control. When the resulting series is truncated and appropriate derivatives taken, The result is:

$$K_T \delta q_i = \delta \lambda_i R - F(q_i, \lambda_i) \quad (2-72)$$

With the introduction of another unknown, the loading parameter, the system of equations is short one equation to be a solvable system. The additional constraint comes from developing a search radius of Δl which gives an arch to look along. The length of the search radius and the additional equation comes from the Pythagorean theorem:

$$\Delta l^2 = \Delta q_{i+1}^T \Delta q_{i+1} + \Delta \lambda_{i+1}^2 R^T R \quad (2-73)$$

The search path is arbitrary and is limited to values small enough to capture the equilibrium curve accurately. The search radius is then approximated by:

$$\Delta l^2 = \Delta q_{i+1}^T \Delta q_{i+1} \quad (2-74)$$

The search radius definition of Eqn(2-74) causes the global stiffness matrix to lose its symmetry. To restore symmetry, δq_i is reduced to two parts:

$$\delta q_i = \delta q_{i1} + \delta \lambda_i \delta q_{i2} \quad (2-75)$$

where

$$\delta q_{i1} = -K_T^{-1} F(q_i, \lambda_i), \quad \delta q_{i2} = K_T^{-1} R \quad (2-76)$$

The updated incremental values of displacement and load parameter can be expressed in terms of the previous increment and a δ change:

$$\Delta q_{i+1} = \Delta q_i + \delta q_i \quad (2-77)$$

$$\Delta \lambda_{i+1} = \Delta \lambda_i + \delta \lambda_i \quad (2-78)$$

Substituting these into the new constraint equation, a quadratic equation results such that:

$$a\delta\lambda_i^2 + b\delta\lambda_i + c = 0 \quad (2-79)$$

where

$$\begin{aligned} a &= \delta q_{i2}^T \delta q_{i2}, \quad b = 2\delta q_{i2}^T (\Delta q_i + \delta q_i), \\ c &= (\Delta q_i + \delta q_i)^T (\Delta q_i + \delta q_i) - \Delta l^2 \end{aligned} \quad (2-80)$$

The above process is iterated until convergence is achieved by defining a limiting value to δq_i . When the (m)th load increment has converged, the total displacement and load parameter values are calculated.

$$q_m = q_{m-1} + \Delta q_m, \quad \lambda_m = \lambda_{m-1} + \Delta \lambda_m \quad (2-81)$$

the search radius for the load step is:

$$\Delta l_m = \Delta l_{m-1} \frac{N_s}{N_{m-1}} \quad (2-82)$$

where N_s is a user predicted number of iterations and N_{m-1} is the number of iterations for convergence on the last load step. Once the values of Eqn(2-82) are found for a load increment, the initial delta load parameter is found from:

$$\Delta \lambda_1 = \pm \frac{\Delta l_m}{\sqrt{\delta q_{i2}^T \delta q_{i2}}} \quad (2-83)$$

2.7 Step by Step Riks Algorithm

The steps shown for Riks were presented by Creaghan [8] and are repeated here for completeness. Figure 2-12 shows the process graphically and should be referenced while following the text description.

1. First increment and first iteration compute only constant stiffness matrix, K .
2. First iteration at each load increment compute $\delta q_{i2} = K_T^{-1}R$. After the first increment, K_T is composed from q_{m-1} nonlinear displacement terms.
3. Each iteration and each increment compute $\Delta q_i = \Delta \lambda_i \delta q_{i2}$. For first increment, $\Delta \lambda_i = \lambda_0$ which is a program input.
4. For each iteration and each increment compute in order:

$$\begin{aligned}\delta q_{i1} &= -K_T^{-1}F(q_i, \lambda_i) \\ \delta q_i &= \delta q_{i1} + \Delta \lambda_i \delta q_{i2} \\ \Delta q_{i+1} &= \Delta q_i + \delta q_i\end{aligned}\tag{2-84}$$

5. Compute $\Delta l = \sqrt{\Delta q_{i+1}^T \Delta q_{i+1}}$
6. Update K_T with $q_i = q_{m-1} + \Delta q_i$
7. Solve quadratic equation of Eqn(2-79) (with given a,b,c definitions) for $\pm \delta \lambda_i$. If roots are complex, return to step 2 and arbitrarily adjust Δl_m , which changes $\Delta \lambda_i$.
8. Choose $\pm \delta \lambda_i$ based on which value yields a positive θ in:

$$\begin{aligned}\delta q_i &= \delta q_i \pm \Delta \lambda_i \delta q_{i2} \\ \theta &= (\Delta q_i + \delta q_i) \Delta q_i\end{aligned}\tag{2-85}$$

if both are positive, then $\delta \lambda_i = -c/b$.

9. Update the displacement and loading parameter by:

$$\begin{aligned}\Delta q_{i+1} &= \Delta q_i + \delta q_i \\ \Delta \lambda_{i+1} &= \Delta \lambda_i + \delta \lambda_i\end{aligned}\tag{2-86}$$

10. Check for convergence. If no convergence for the increment yet, return to step 2.

Upon convergence, update displacement and loading parameter for next increment:

$$\begin{aligned} q_m &= q_{m-1} + \Delta q_m \\ \lambda_m &= \lambda_{m-1} + \Delta \lambda_m \end{aligned} \quad (2-87)$$

11. Compute new search radius for next increment:

$$\Delta l_m = \Delta l_{m-1} \frac{N_m}{N_{m-1}} \quad (2-88)$$

where N_m is user provided iteration estimate and N_{m-1} is the number of iterations required for convergence in the last increment.

12. Compute loading parameter for first iteration of the next increment:

$$\Delta \lambda_1 = \pm \frac{\Delta l_m}{\sqrt{\delta q_{i2}^T \delta q_{i2}}} \quad (2-89)$$

13. Return to step 2 for next load increment.

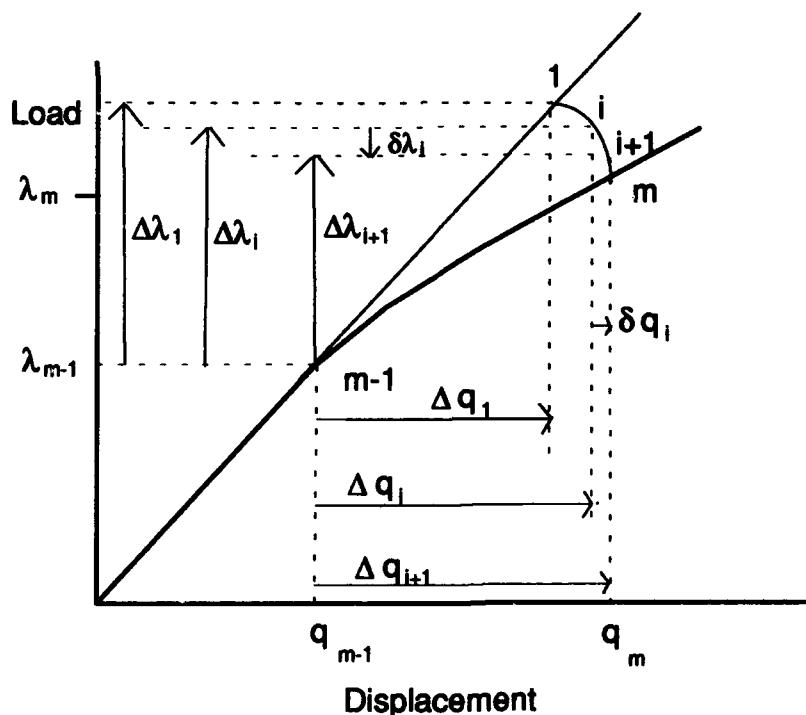


Figure 2-12 Riks Technique

2.8 Arch Geometry Definitions

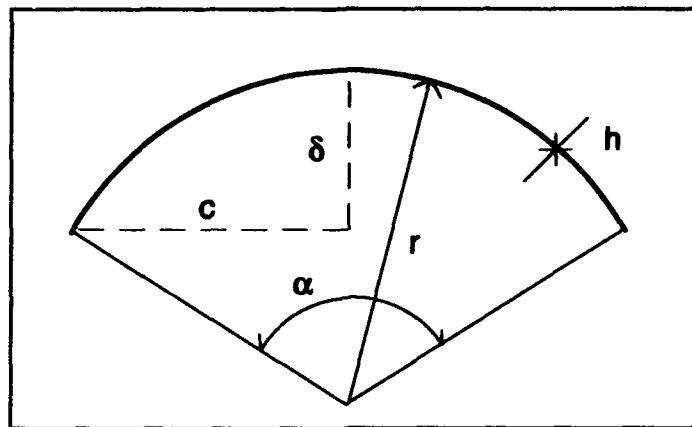


Figure 2-13 Typical Arch Geometry

Before moving to specific problems, it is necessary to establish criteria to categorize arches. Each arch will be categorized as deep or shallow and thick or thin. A thick arch is defined as one for which through-the-thickness shear becomes significant. Through-the-thickness shear becomes significant when solutions with and without shear differ by 10 %. Huang [16] implies that thin arches have $h/r \ll 1$. This definition is inadequate for the current problems. An arch could have a large radius of curvature, but have a small arc length. In this case, Huang would define an arch as thin when shear is a major factor. In fact, flat beams would all be classified as thin. Our definition needs to include a dimension to account for the arc length as well as the thickness. The definition we chose is somewhat arbitrary, but since it will only be used to categorize problems, it will not affect solutions. We will consider an arch thick when:

$$\frac{c}{h} < 25 \quad (2-90)$$

or a flat beam is considered thick when:

$$\frac{l}{2h} < 25 \quad (2-91)$$

where l is the length of a flat beam.

The second geometric parameter for classifying an arch is depth. Depth is a measure of how far removed an arch is from a flat beam. Smith [33] states that a shallow shell is one with $\delta/c \leq 0.3$. This definition would work well for the current problems, but we seek a depth definition that incorporates the opening angle in a more direct way. Huang [16] defined a depth parameter, λ , by:

$$\lambda = 24\sqrt{3(1-v^2)} \left(\frac{\delta}{h}\right)^5 \quad (2-92)$$

Eqn(2-92) is limited to $\alpha \ll 1$. Many of the deep problems have large opening angles making this definition unusable. Fung [14] uses Eqn(2-92) but removes the angle limitation such that:

$$\lambda = 4\sqrt{12(1-v^2)} \left(\frac{r}{h}\right)^5 \frac{\alpha}{2} \quad (2-93)$$

Fung shows that δ/h is an appropriate depth parameter for shallow shells, but that Eqn(2-93) is appropriate for deeper shells. We adapt this definition directly to arches. We will use the criteria that if $\lambda \geq 8$, then the arch is considered deep. This value is arbitrary and was chosen to match problem classification in other publications. Since the problems analyzed here will be loaded in bending, the laminate value of v_{21} should be used in place of v for a composite material. These definitions are used in chapter III to classify the problems so that the reader can get an idea as to the complexity of each problem before examining each in depth.

III. Discussion and Results

3.1 Clamped Isotropic Shallow Thin Arch

The first family of problems involve large displacements and geometric nonlinearity, but do not experience extremely large rotations due to bending. These first cases are used to show that the new large rotation considerations haven't corrupted previous results. Under small bending rotations (ψ less than about 20 degrees), the current results should match Creaghan's [8] closely. These initial problems will also show the validity of the current theory on previously verified problems. The first problem considered is an isotropic arch with two clamped boundary conditions loaded by a single concentrated load at the crown. The exact problem geometry and properties are shown in figure 3-1. The problem was examined by Belytschko and Glaum [4] and Creaghan [8]. Each attained equilibrium solutions past geometric collapse or snap. Creaghan showed comparisons using small bending angles to the 2-D simplified large displacement/rotation (SLR) theory of Palazotto and Dennis [27]. No differences were observed between Creaghan and the SLR theory, so comparisons here are made with Creaghan and Belytschko and Glaum only.

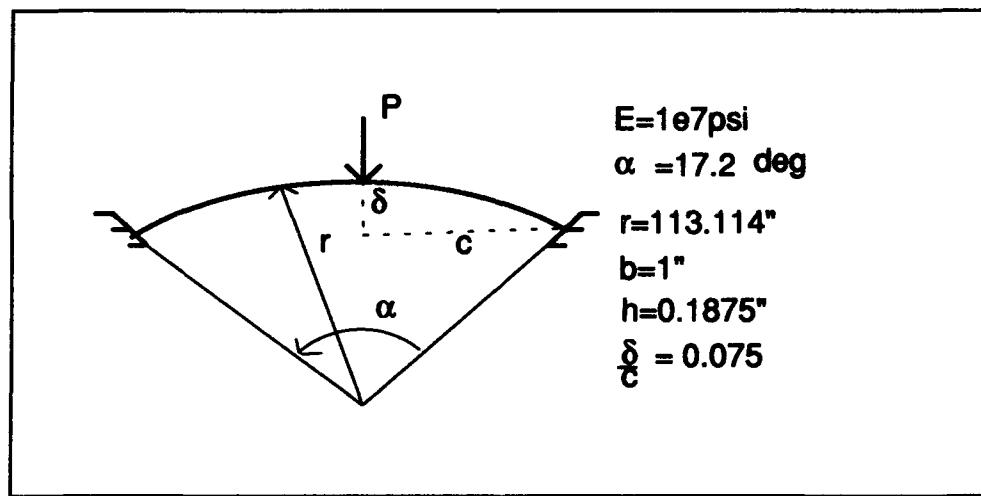


Figure 3-1 Clamped Clamped Shallow Thin Arch

The problem was analyzed with a symmetric half arch model with 20 elements and 96 active degrees of freedom. As seen in figure 3-2, there is little or no discernible difference between Creaghan's solution and the present solution. Even though the arch experiences vertical displacements in excess of ten times its thickness, there is bending rotation of only 10 degrees (as measured over 20% of the beam). The small angle theory used in Creaghan's bending kinematics is a 99% accurate approximation of the tangent at 10 degrees. This first result is an initial confirmation that the present theory hasn't introduced any gross errors in the program.

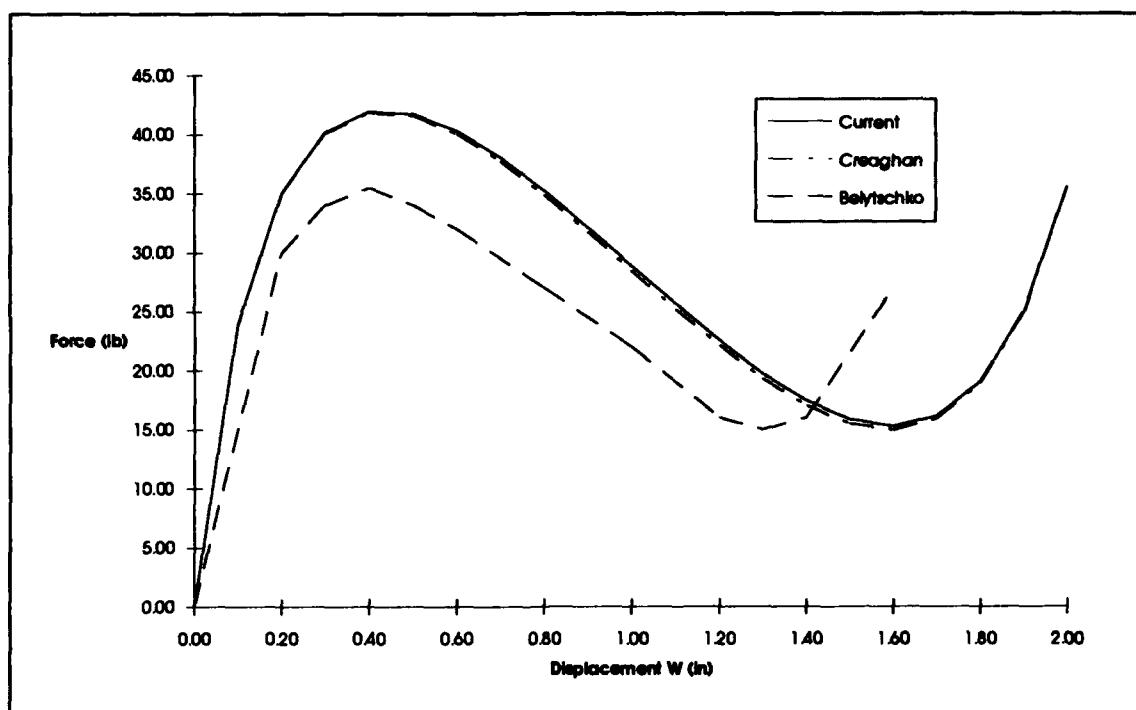


Figure 3-2 Comparison for Clamped Clamped Shallow Thin Arch

Belytschko and Glaum use an updated Lagrangian corotational coordinate formulation. They embed a coordinate system in an element to track average rigid body motion during deformation. Unlike a total Lagrangian formulation, their displacements

are divided into rigid body motion and real structural deformation. Belytschko and Glaum use standard Euler-Bernoulli beam assumptions so that cross sectional normals remain straight and normal (no warping due to transverse shear). They also neglect many in-plane nonlinear strain expressions. Using the current nomenclature, their mid-plane strain can be expressed as:

$$\epsilon_2^o = v + \frac{1}{2}(w_{,2})^2 + \text{rotation term} \quad (3-1)$$

where the rotation term relates the corotational axes and the arch center line. Since the current formulation is a total Lagrangian, there is no comparable term. The present work retains more higher order mid-plane strain terms as shown in Eqn(2-31) for ϵ_2^o .

The half arch analysis using the current model shows a stiffer structure initially than Belytschko and Glaum. Figure 3-2 also shows Belytschko and Glaum predicting a lower collapse load than Creaghan and the present work. The higher order in-plane strain terms accounts for the stiffer initial response and higher collapse load. The current structure is able to absorb more energy before collapse because of the inclusion of higher order strain terms. After structural snap, both theories unload to nearly the same force, but at different displacement values. This post collapse softening is common for other higher order shell theories when compared to the simpler counter parts. Palazotto and Dennis [27] showed similar behavior when evaluating nonlinear arches and comparing them to lower order theories. They attribute this trend to higher order mid-plane strain representation.

3.2 Clamped Isotropic Shallow Thick Arch

The second problem is another shallow arch with two clamped end supports and a concentrated load at the crown. This problem is moderately thick with c/h of 12. The thicker arch implies more significance of the through-the-thickness shear strain. This

problem just meets the criteria for a thick arch, but is still considered shallow as the arc length is short (49 inches) for the radius of curvature (100 inches), resulting in a depth parameter (λ) of 6.3. For this problem, a larger opening angle would be necessary before categorizing the arch as deep.

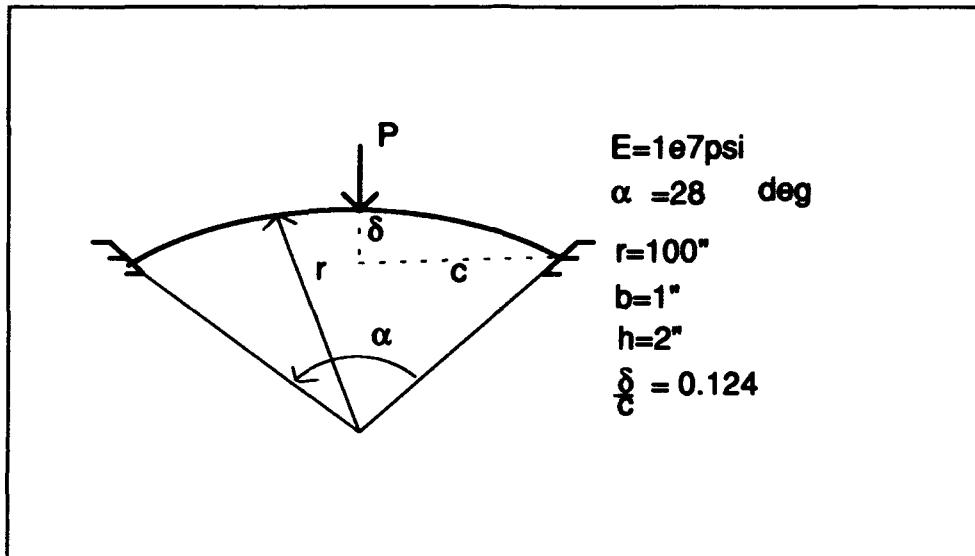


Figure 3-3 Clamped Clamped Shallow Thick Arch

This problem was also examined by Belytschko and Glaum [4]. They show an exact solution to this bending problem but subject to Donnell assumptions. The Donnell assumption in their solution means that membrane (mid-plane) rotation effects are neglected. The present model involved a half symmetric arch with 20 elements and 96 active degrees of freedom. The result shown in figure 3-4 shows a very good comparison with Belytschko and Glaum.

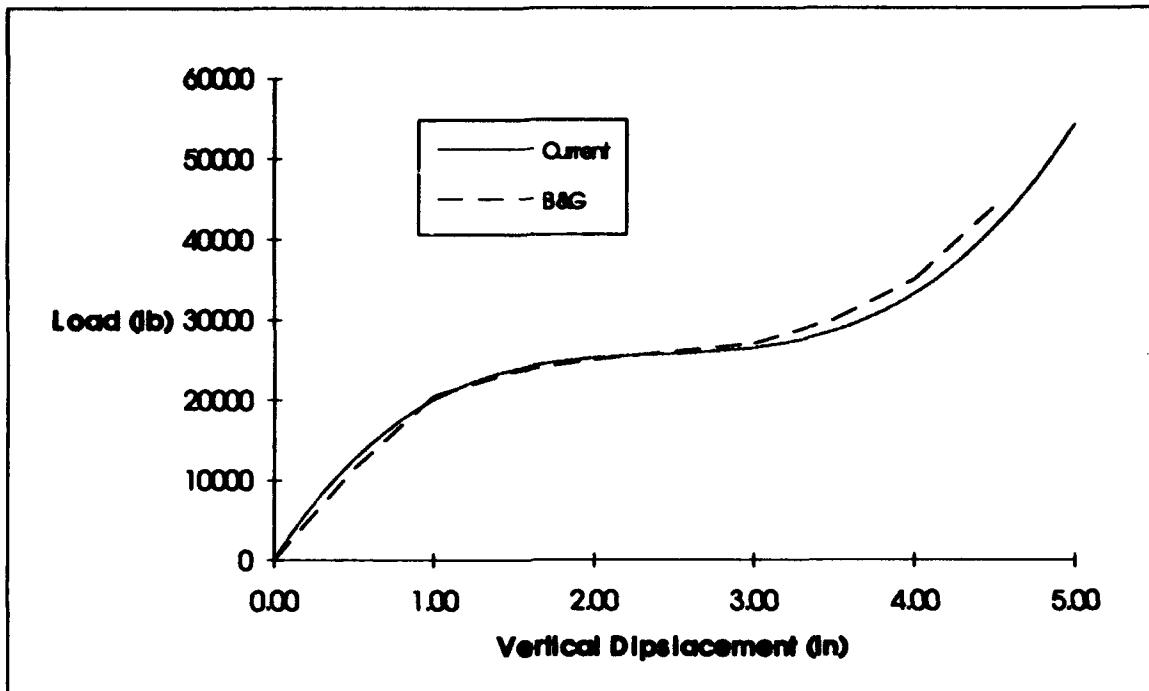


Figure 3-4 Comparison for Clamped Clamped Shallow Thick Arch

Even though the Donnell solution includes shear, this problem acts more like a flat beam than the previous problem. Since the load displacement curve doesn't exhibit a limit load, there is no instability or snapping as in the first arch. At the point of geometric inflection, the slope of the load displacement curve changes, but the load never decreases. The entire loading path is stable. A continuously increasing load would be characteristic of a flat beam while an inflection point is a curved arch characteristic. This problem is shallow and thick enough to act more like a flat beam than like an arch. If this arch were deeper, there would be more geometric instability evidenced by snapping. The arch experienced small to moderate bending rotations of 16 degrees. This problem again is a good demonstration of the code for a moderately difficult problem.

3.3 Clamped Isotropic Thin Straight Beam

The final problem examined in the small rotation area is a flat beam clamped at both ends with a concentrated transverse load at the center. The beam is isotropic with geometry details shown in figure 3-5. Mondkar and Powell [23] examined this problem using a Lagrangian formulation of the static and dynamic response. Strain is formed in a Lagrangian system and includes nonlinear terms. Mondkar and Powell use a load incrementing Newton-Ralphson iteration scheme similar to the current effort.. They neglected all out of plane displacement by setting Poisson's ratio to zero.

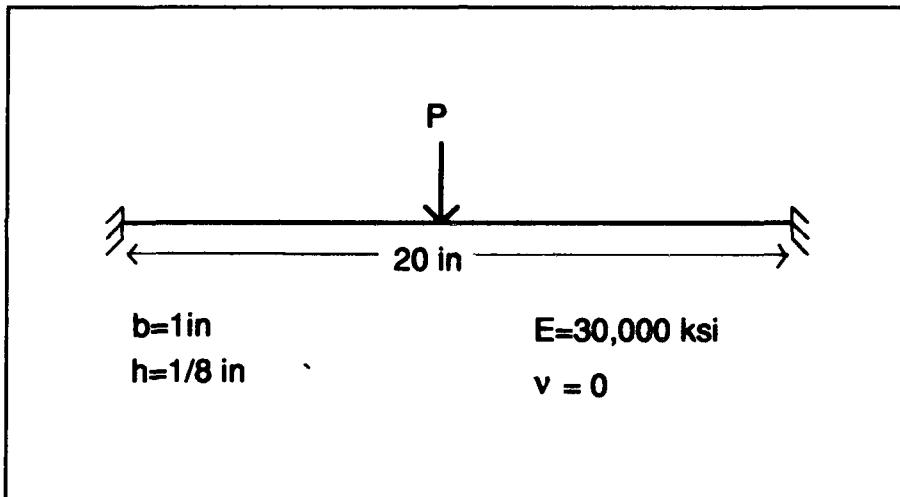


Figure 3-5 Clamped Clamped Flat Thin Beam

The beam was modeled using a symmetric half width beam with 20 elements and 96 active degrees of freedom. Displacement control was used to find the equilibrium solutions which are displayed in figure 3-6. For stable problems like this, large displacement increments are possible with little variation in the overall results. The solution shown in figure 3-6 used .05 inch displacement increments which is rather large

when compared to the thickness. Since there are no instabilities, Riks technique gives identical results and predicts only one equilibrium point for a specific load. Figure 3-6 shows the behavior of a flat beam with a continuously increasing load displacement curve with no geometric inflection. The Mondkar and Powell data is shown as specific points which are equilibrium displacements they found for various loads. The vertical displacements are moderate at a maximum of four times the thickness and small bending rotations of 4 degrees. The results compare well with the Mondkar and Powell solution. The rotations are small, but this adds confidence to the accuracy of the code for simple nonlinear problems.

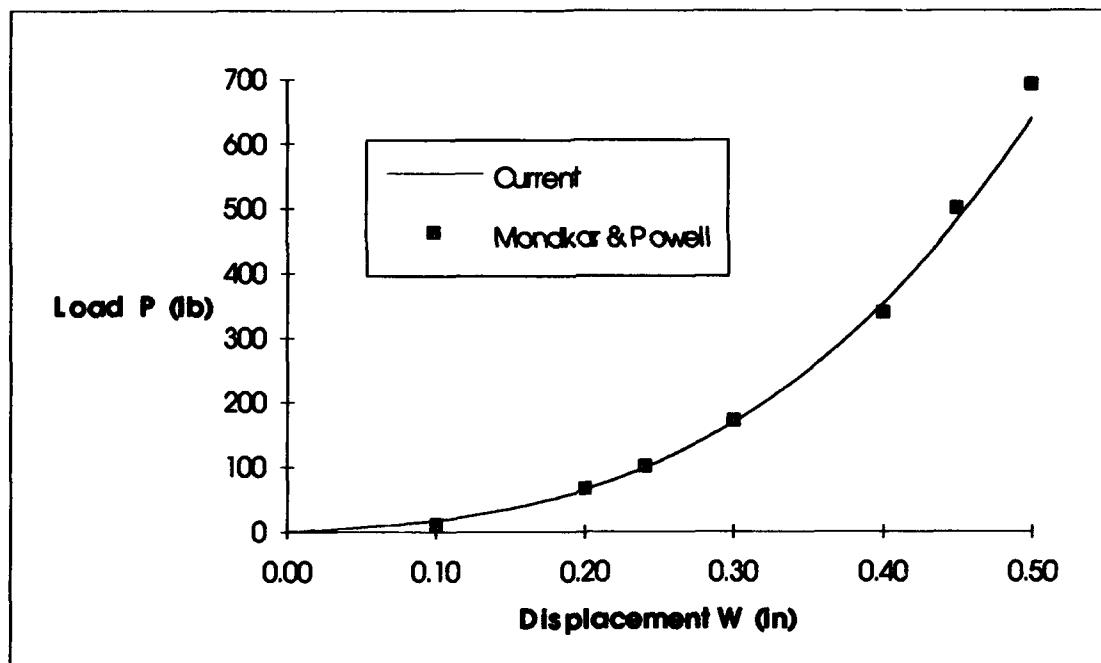


Figure 3-6 Equilibrium Curve and Comparison for Clamped Clamped Thin Beam

This problem concludes the initial family of small rotation problems. The present theory provided reasonable results with little difference (except for Belytschko and Glaum solution in section 3.1) over small angle kinematic theories. Besides comparing well with

other published data, these problems provided a good initial check of the theory and its implementation in the code. The next group of problems is designed to give more geometric nonlinearity than the first three with good solutions at large bending angles as the final goal.

3.4 Cantilever Isotropic Thin Beam

The first large rotation problem examined is a cantilevered isotropic straight beam subjected to a concentrated transverse end load. Figure 3-7 shows the problem geometry and material properties used.

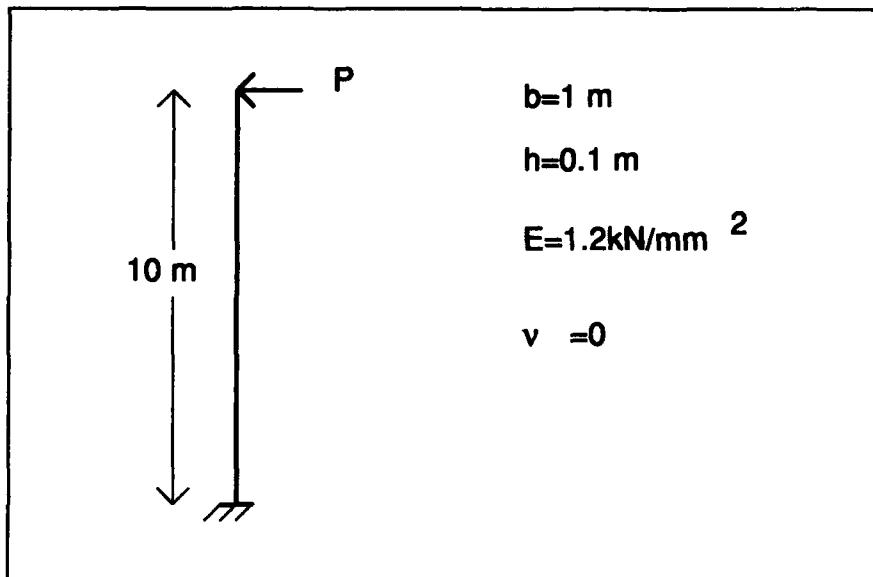


Figure 3-7 Cantilever Isotropic Beam

This problem was examined by Hsiao and Hou [15]. Their work centered on removing small rotation restrictions between increments in an updated Lagrangian formulation. They utilize a corotational formulation system in which rigid body motion and deformations are separated by attaching a coordinate system to an element during loading. The current work is to incorporate large bending rotations in a total Lagrangian system making their updated Lagrangian large rotation effort of particular interest. Hsiao and Hou base their analysis on three assumptions:

1. Euler-Bernoulli assumptions are valid (plane sections remain plane - no warping).
2. Membrane strain (strain at the beam mid-plane) is constant along the beam.
3. Axial strain is small.

The first two assumptions differ from those used in the current theory. The first assumption neglects through-the-thickness shear strain. However, since this beam is thin ($l/2h = 50$), transverse shear should be small. Their second assumption of constant membrane strain is also in conflict with the present theory. Eqn(2-30) and Eqn(2-31) show in-plane strain which includes many higher order terms such that the membrane strain varies along the length and through-the-thickness. The third assumption Hsiao and Hou make is that the strains are small which is compatible with the present work. This assumption allows the linear elastic constitutive relationships for stress and strain to be used.

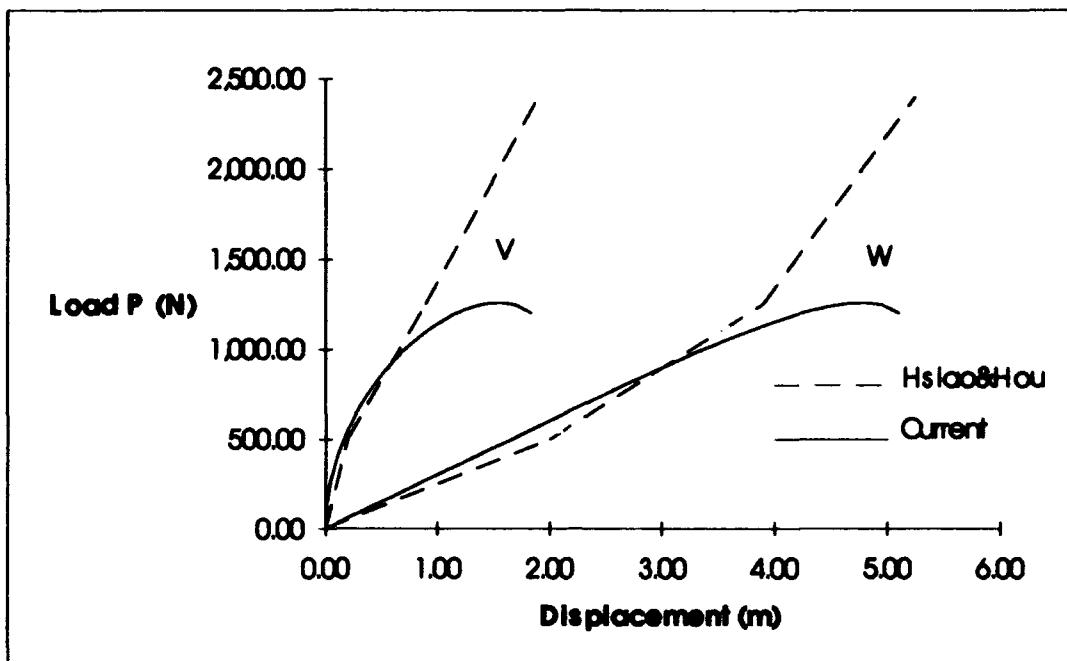


Figure 3-8 End Displacement for Cantilever Isotropic Beam

The analysis utilized 40 elements (200 active degrees of freedom) and displacement control with displacement increments of the cantilever end of 0.15 inches. Figure 3-8 shows the current theory correlating well with Hsiao and Hou up to 4.2 meters of vertical end deflection. This displacement is 42 times the beam thickness making this a highly nonlinear solution. The present solution diverges from Hsiao and Hou between 4.2 and 5.0 meters of end displacement. Upon examination of bending rotations in that area, one finds that ψ goes from 37 degrees at $w=4.2$ m to 58 degrees at $w=5.0$ m. In the range of bending angles over approximately 50 degrees, the assumption on constant vertical displacement through-the-thickness causes in-plane displacements to get large as shown in chapter II due to incorporation of the tangent function. As ψ approaches 90 degrees, the in-plane displacements become infinite. This explains the inconsistency past 4.2m of deflection. The solution is within 10% of the Hsiao and Hou solution up to 40 degrees of

bending rotation. This problem gives the first indication as to the bending limits of the current theory. The structural softening exhibited at large rotation was experienced by Creaghan [8], but at a much lower rotation of 24 degrees. It will be shown in sections 3.6 and 3.10 that including the tangent function in the kinematics has pushed this softening to higher load and displacement values (as compared to Creaghan) making it a valuable addition.

3.5 Cantilever Isotropic Beam with Tip Moment

This problem is a thin isotropic cantilevered beam with an external end moment. Figure 3-9 shows this beam which was analyzed by Epstein and Murray [12]. No cross section dimensions were specified, so a rectangular geometry was used with the dimensions to satisfy the area and moment of inertia requirements. The resulting beam is very thin (0.0346 in) and wide (28.9 in).

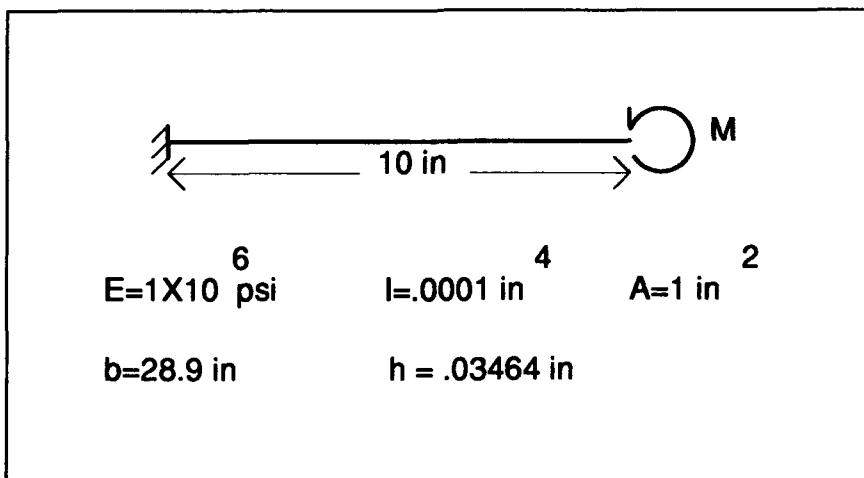


Figure 3-9 Cantilevered Isotropic Beam with Tip Moment

Epstein and Murray use a corotational total Lagrangian formulation with major beam theory simplifications. They neglect through-the-thickness shear strain, but it should not be significant for this problem because the cross section is thin. They retain only one strain component which can be expressed using the current notation as:

$$\epsilon_2 = (1 - \kappa z)^2 \left(v_{,2} + \frac{1}{2} (v_{,2}^2 + w_{,2}^2) \right) - \frac{1}{2} \quad (3-2)$$

where κ is the beam curvature. The last term of Eqn(3-2) [1/2] is a result of finding the Green strain with the no warping assumptions they used. Epstein and Murray use the internal virtual work principle while keeping the curvature exact (even for large strains). The physical curvature, κ , is defined as the rate of change of the mid-plane normal vector while moving along the beam. They make no limiting approximations outside of the beam assumptions. The resulting equilibrium equations are:

$$\begin{aligned} T \cos(\phi) + S \sin(\phi) &= (2e + 1)N \\ -T \sin(\phi) + S \cos(\phi) &= \frac{-(2e + 1)M}{\sqrt{2e + 1}} \end{aligned} \quad (3-3)$$

where T is an externally applied axial force, S is an externally applied transverse force, M is a resulting internal bending moment, N is an internal resulting force, ϕ is the rotation angle of the mid-plane tangent at an end and e is the Green strain of the mid-plane ($z=0$ in Eqn(3-2)). Epstein and Murray use exact trigonometric functions to describe bending angle effects which enabled them to wrap the beam into a complete circle. The biggest limitation comes from the simple beam approximations. The current theory is higher order in strain, but uses geometric approximations and assumptions to simplify the equations.

The resulting moment was calculated using:

$$M = -EI\psi_2 \quad (3-4)$$

where the shape functions of Eqn(2-58) were used to find ψ_2 at the beam end. Figure 3-10 shows the comparison between the present theory, Creaghan and Epstein and Murray for vertical deflection, while figure 3-11 shows the comparison between current results and Epstein and Murray for horizontal displacement. Creaghan's data was post processed using Eqn(3-4) with the results shown in figure 3-10.

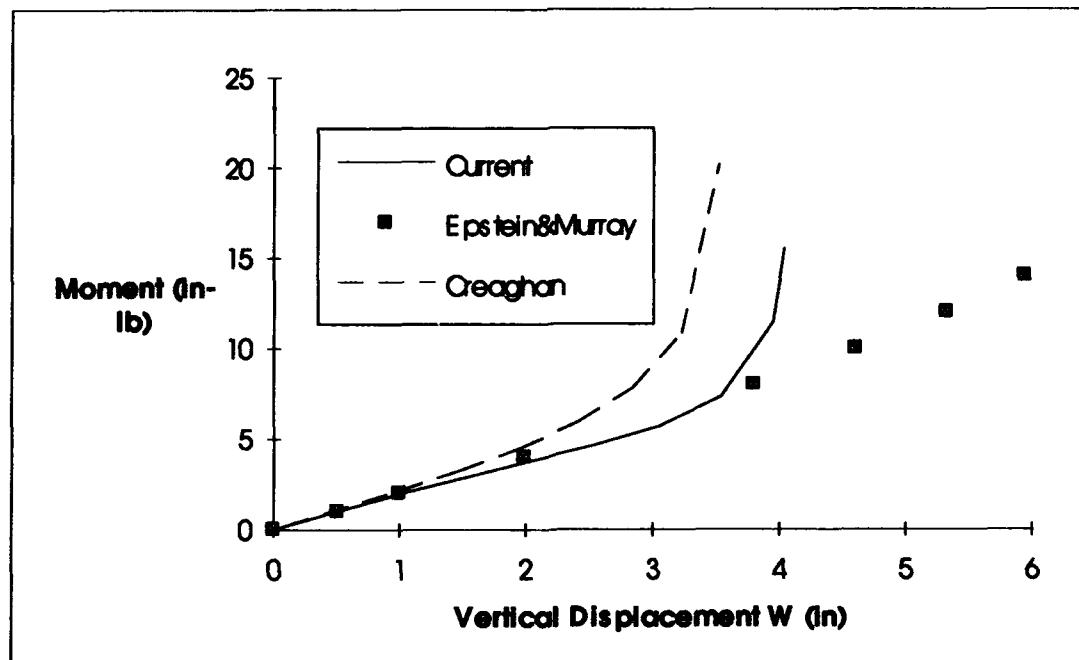


Figure 3-10 Cantilever Beam with Tip Moment - Vertical Displacement

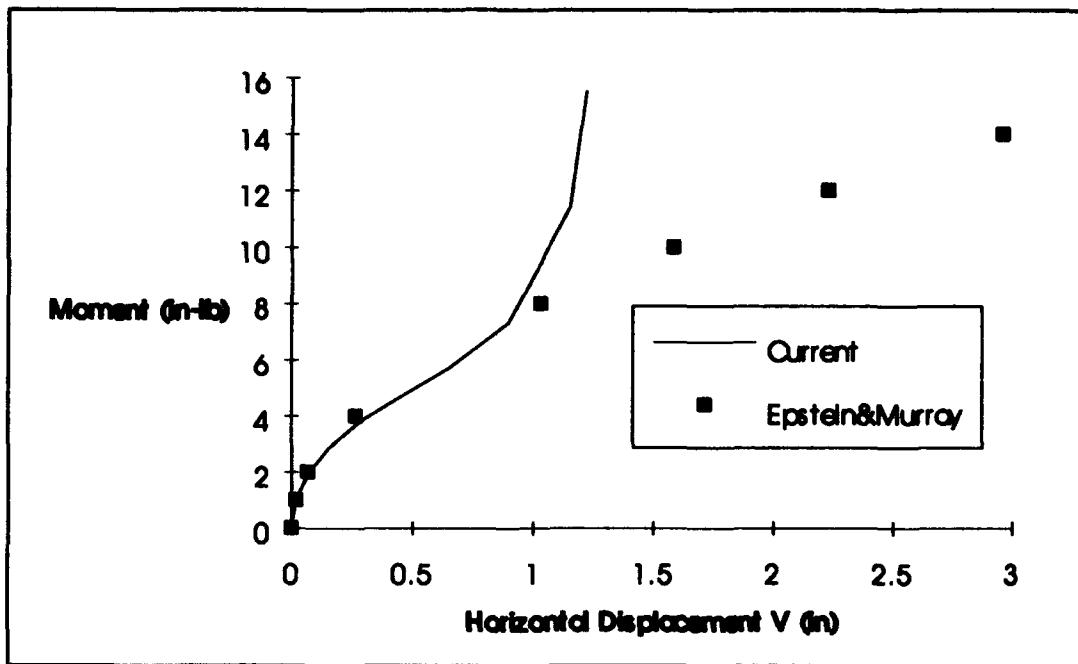


Figure 3-11 Cantilever Beam with Tip Moment - Horizontal Displacement

Creaghan showed initial divergence from Epstein and Murray at 5 in-lb while the current solution starts to diverge at 9 in-lb of bending moment. At this point, the bending rotations are 40 degrees over 20 percent of the beam end. This is nearly the same rotation angle where the previous problem diverged. The horizontal displacements compared well until 7 in-lb, where the current solution becomes stiffer than Epstein and Murray. The present solution follows the same trend as Creaghan, but stays flexible to higher displacements.

3.6 Cantilevered Composite Beam with End Load

This problem consists of a cantilevered composite beam with a concentrated transverse load applied at the free end. The rectangular beam is constructed of layered lamina made from AS4-3501-6 graphite epoxy in various orientations. This problem is of particular interest as Minguet and Dugundji [22] present actual test data from a composite helicopter rotor blade test and analysis program. Minguet and Dugundji use an updated Lagrangian formulation based on Euler angles to track element rigid body motion relative to the original coordinate system. They include constant transverse shear strain, mid-plane extensibility, and axial/shear coupling, making their theory close to the present and more advanced than most other beam theories. As already discussed, the current work is limited to balanced symmetric composite lay ups, therefore, the $[0/90]_s$ laminate is analyzed. The lamina material properties are: $E_1=142$ GPa, $E_2=9.8$ GPa, $G_{12}=G_{13}=6$ GPa, $G_{23}=4.8$ GPa, and $\nu_{12}=.3$. The direction designators here correspond to the prime system in figure 2-8 (material axes).

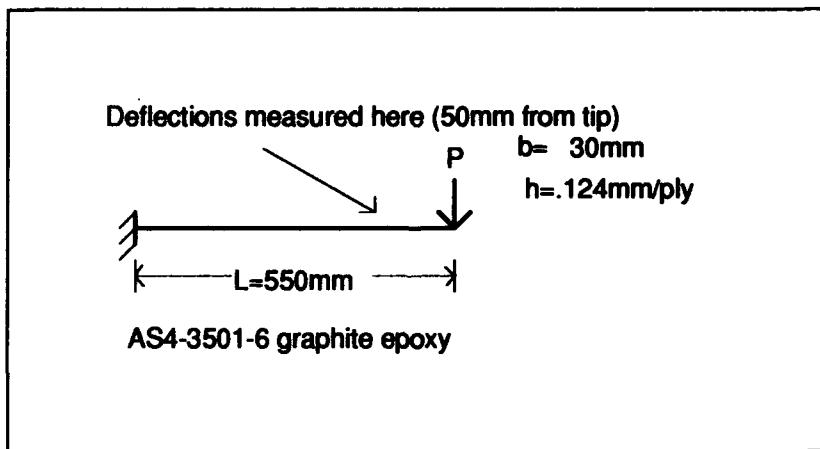


Figure 3-12 Cantilever Composite Beam

Minguet and Dugundji tested the beam in figure 3-12 by stacking various weights on the tip and measuring vertical and horizontal displacements at a point located 50mm from the beam end. The results in figure 3-13 and figure 3-14 show data points from the actual

tests. The problem was analyzed using a full beam model with 33 elements and 165 active degrees of freedom. Displacement control was used to increment tip vertical deflection and iterate to load equilibrium solutions.

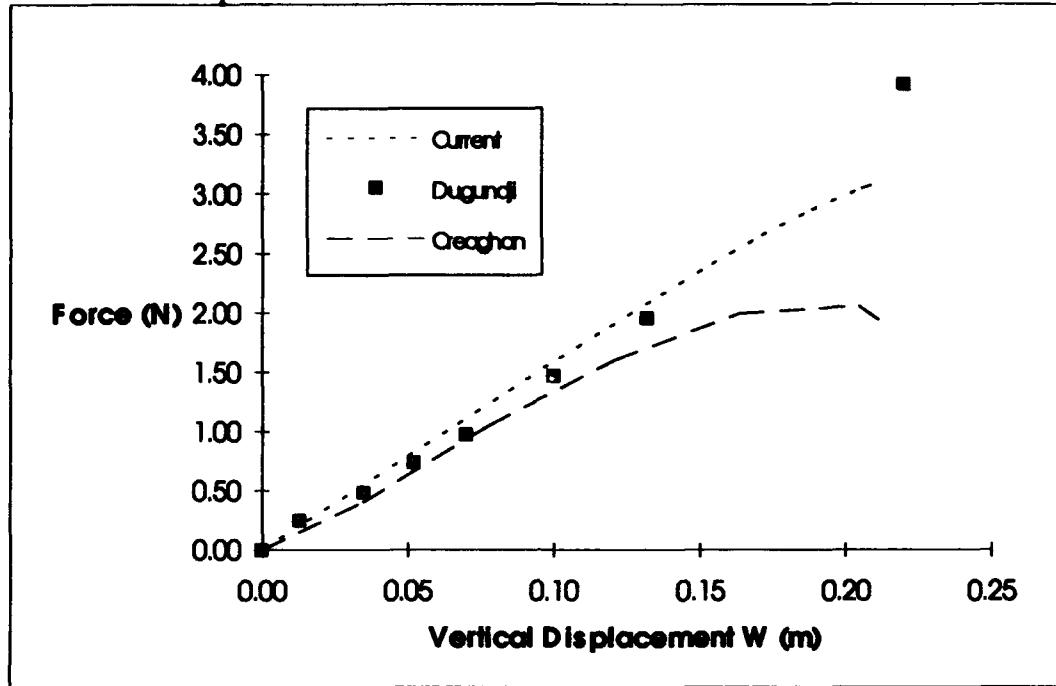


Figure 3-13 Vertical Displacement Comparison for Cantilever Composite Beam

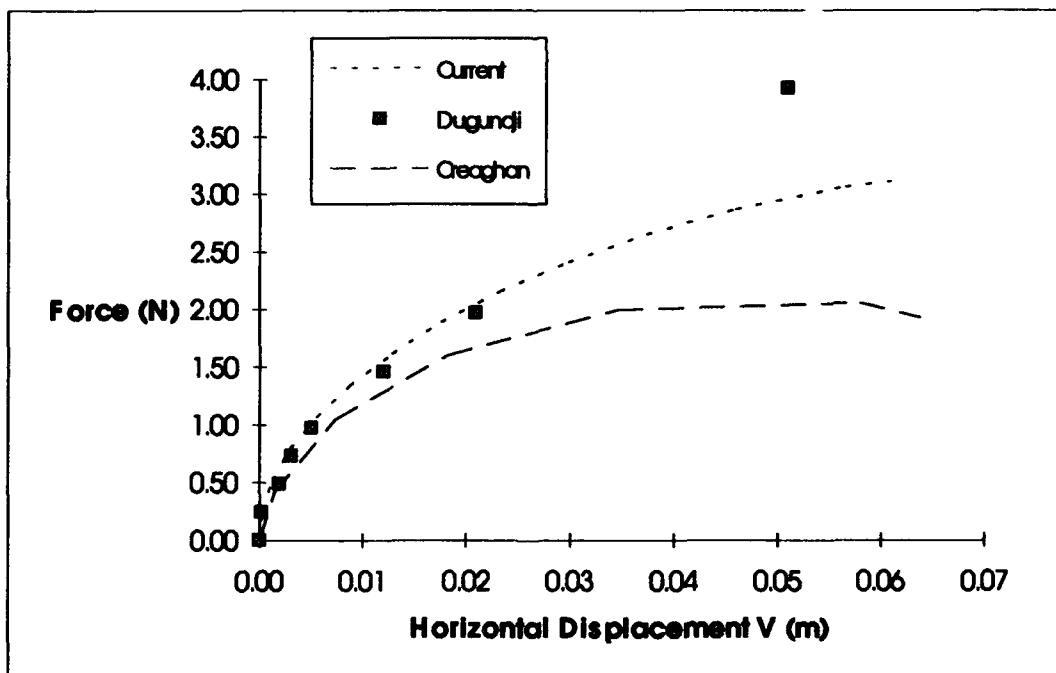


Figure 3-14 Horizontal Displacement Comparison for Cantilever Composite Beam

As seen in figures 3-13 and 3-14, the present theory fits the test data much more accurately than before large rotation considerations were included. The current results closely match the test data all the way to vertical deflections 134 times the laminate thickness. Creaghan showed 10% error at 0.15m of vertical deflection while the current solution shows improvement to 10% error at 0.22m of vertical displacement. At 40 degrees of bending rotation, the present results differ from the test data by 15% while Creaghan differed by 50% at the same place. This shows a significant solution improvement by incorporating the tangent function in the kinematics. All of the current results compare well with test data until application of the 400 gram weight. At this load, the current model experiences structural softening again. There appears to be a point in these problems where the bending angle relationships start to break down. The softening is much more pronounced in the horizontal displacement results. All of the large rotation problems so far start to exhibit incorrect behavior beyond 40 degrees of bending rotation. Each of the large rotation problems presented so far have had at least one end that is unconstrained. As will be shown later in this chapter, problems with more bending constraints at the boundaries will give good solutions for about another 5 degrees of rotation. This problem proved to be an excellent test for the current formulation by comparing directly to actual test data of a composite beam.

3.7 Hinged Isotropic Shallow Thick Arch

The next problem is a shallow thick isotropic arch with both ends pinned. It has been evaluated by Sabir and Lock [31] and Creaghan [8]. This arch is interesting because it experiences snap through and snap back limit points. As displacement control cannot capture snap back characteristics (section 2-6), a Riks solution is more useful to capture

limit points and possible equilibrium paths. The arch is relatively thick ($c/h = 14.5$) making transverse shear significant. The arch details are shown in figure 3-15.

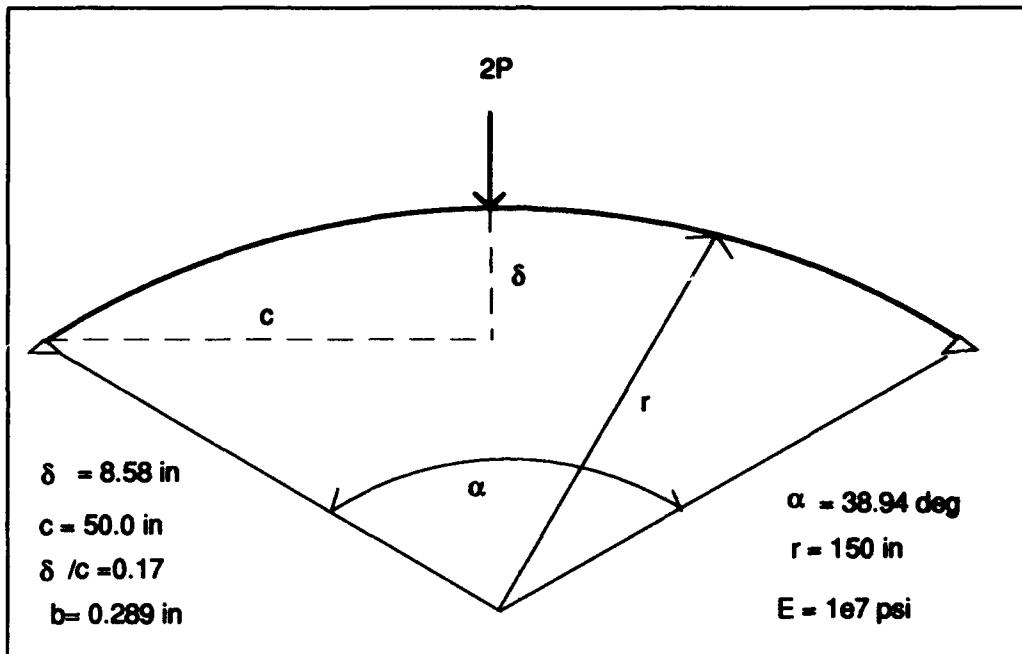


Figure 3-15 Hinged Hinged Shallow Thick Arch

Sabir and Lock have a more advanced representation for ϵ_2 than most of the other theories. They represent longitudinal strain as:

$$\epsilon_2 = v_{,2} + \frac{w}{r} + \frac{1}{2} w_{,2}^2 - z \left(w_{,22} - \frac{v_{,2}}{r} \right) \quad (3-5)$$

Eqn(3-5) contains a thickness dependence for the in-plane strain. The current theory contains many higher order z strain terms, while others neglect these terms totally. Sabir and Lock also neglect transverse shear strain which is present in this problem.

The results for a half arch symmetric model using Riks solution technique are presented in figure 3-16. Analysis points from Sabir and Lock are shown. No attempt was made to connect their points for the simple reason of chart readability. The current solution

follows Creaghan and Sabir & Lock through four horizontal and two vertical limit points. As mentioned, Sabir and Lock neglect transverse shear which explains the slightly softer solution at larger displacements (point C). At point C, transverse shear strain is around 2 degrees. As shown in chapter II, the slope of the elastic curve w_2 and the bending angle ψ differ by this shear angle β . This deformation contributes to the overall displacement making the present solution slightly softer.

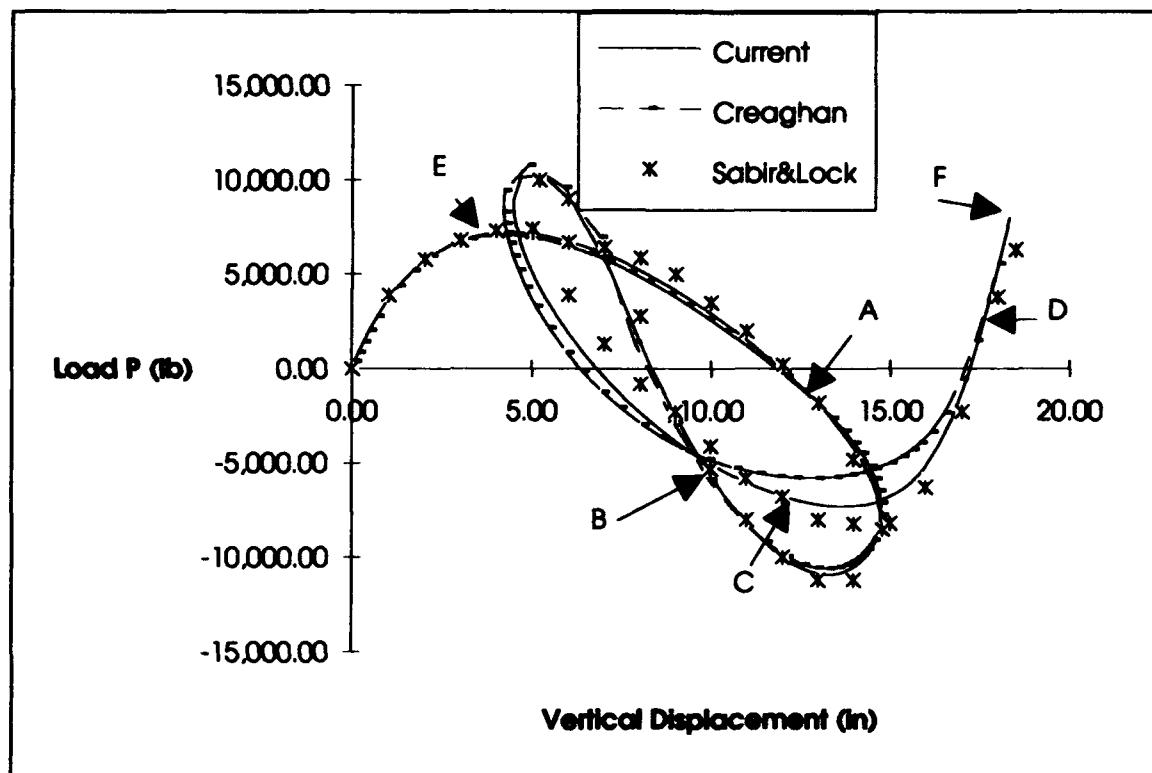


Figure 3-16 Equilibrium Path for Hinged Hinged Shallow Thick Arch

Figure 3-16 doesn't necessarily represent a physically realistic static load displacement path, rather it shows possible equilibrium states even during dynamic events (snapping). Classical load control could not produce such a curve as more than one displacement

exists for a single load. The same is true for the displacement control except there is more than one load for a given displacement. Load control would jump from point E to F instantaneously. Displacement control would follow a path along points E A D without capturing the other points. This problem was solved using displacement control and the results are shown with Riks method to demonstrate this effect. It is evident in figure 3-17 that the snap back points are not captured by displacement control.

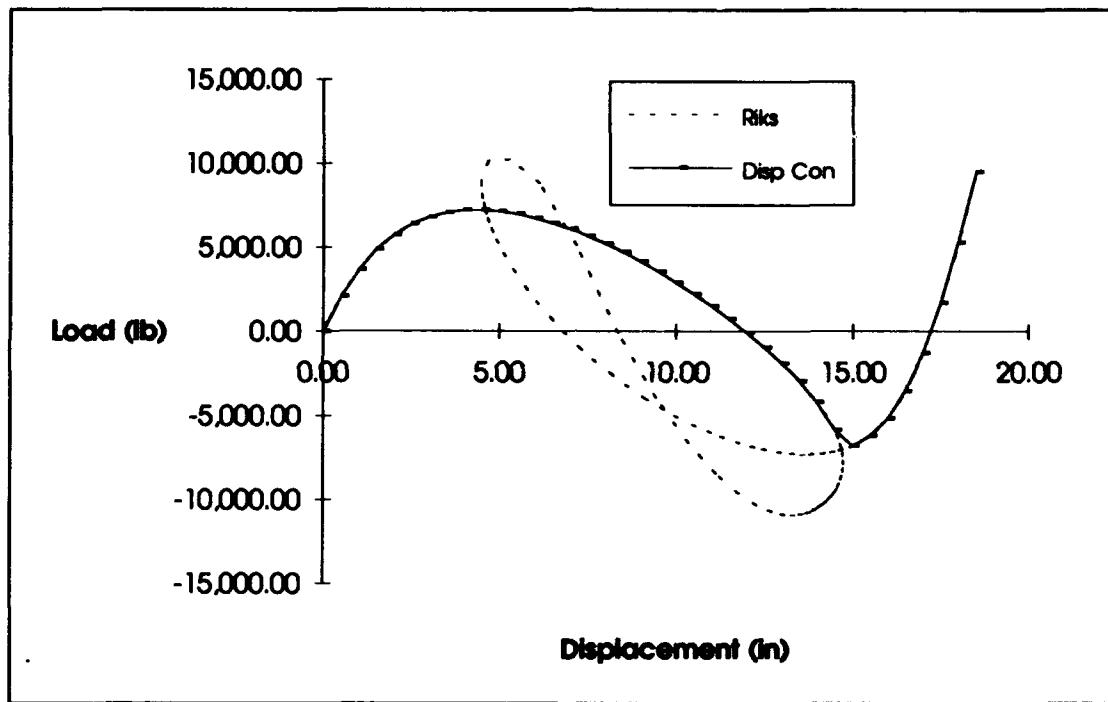


Figure 3-17 Displacement Control vs Riks Method

The loading path in figure 3-16 and deformed shapes in figure 3-18 provide a better understanding of the physical meaning of such a strange equilibrium curve. As the arch goes through its first snap at horizontal limit point E in figure 3-16, the structure collapses and the load decreases until the sense changes to an upward load. Point A shows when

this upward loading begins. As we continue around the curve toward point B, the equilibrium curve shows load and deflection decreasing. The deformed shape of B in figure 3-18 confirms that the crown deflection is less than for point A. This unloading is a dynamic phenomena which occurs nearly instantaneously as the arch collapses and inverts. As displacement is decreasing, the magnitude of the upward force to attain equilibrium decreases also until the second horizontal limit point. The area near point B is typical showing increasing load with decreasing displacement. The process is once again reversed through another snap and sheds load until point C. Now the structure once again becomes stable and will start to behave with positive displacements with positive loads. At point D, all the curvatures are reversed from the original configuration making it stable in an inverted state. Note, the deflected shapes are not drawn to scale. Although the problem curvature has been exaggerated to more easily show load effects, their relationship to the original shape is correct.

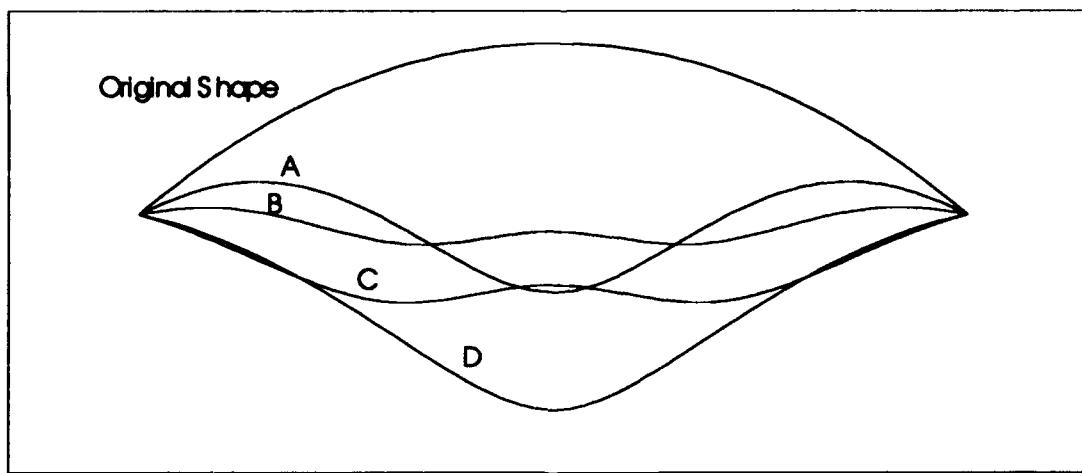


Figure 3-18 Hinged Hinged Shallow Arch Deformed Shapes

Since this problem has centered on Riks solutions, it is appropriate to briefly discuss the practical application of Riks method. The technique can be difficult to correctly get

started for a particular problem, but once a Riks solution is achieved, the results are generally excellent. The easiest way to start a new Riks solution is to first use a displacement control solution. This provides an indication of the loads and some of the snap or limit points that may exist. Riks technique is sensitive to loading parameters. It is necessary to have accurate load and load constants for good Riks solutions. Displacement control shows the load displacement trend and magnitudes of each. In areas of an equilibrium curve that can be captured by both techniques, the results are nearly identical. Although the estimate of iterations needed per increment, N_i , is an integer, in practice, using a value that is not a round number works better. N_i is usually at least 2 for problems that converge easily, such as the current problem where N_i was 2.5. N_i could be as large as 4 or 5 for more difficult problems, but it is rarely higher. N_i was 4.5 for a deep arch problem in section 3-10. M_i is the maximum load increment for a particular iteration. This is a good parameter to vary significantly for difficult problems. Displacement control solutions will also help select an initial value of M_i . Displacement control will help determine the size of the external load(s) to apply for a Riks solution. Too large or small of an external load will provide poor or nonconvergent solutions. A reasonable external load is critical to a good Riks solution. The final significant parameter for a good Riks solution is the convergence tolerance, generally 0.01, but it can be varied to help a particular solution converge. Care should be taken not to loosen this tolerance too much as solutions can converge to incorrect values only to diverge at a later increment. The initial load parameter (λ_0) is generally set to 0.1. Solutions don't appear to be sensitive to this, but it could be varied if getting convergence on the first increment is difficult. The above ideas are only suggestions based on the author's experience. Basically, the author suggests getting a general understanding of the loads and displacements, then vary Riks specific parameters. Varying the load and Riks parameters at the same time can prove

frustrating by giving incorrect solutions for seemingly no reason. Because every problem is different, experimentation is necessary when using Riks technique.

3.8 Deep Hinged Isotropic Thin Arch

The next problem is designed to produce large displacements and large bending rotations. The deep arch in figure 3-19 has two pinned ends and is loaded transversely by a concentrated load at the crown. The arch has been investigated by Huddelston [17], Palazotto and Dennis [27] and Creaghan [8]. Creaghan made comparisons to SLR shell theory of Palazotto and Dennis [27]. The differences were negligible, so the current theory will be shown with Creaghan beam solutions, Huddelston, and a Donnell shell analysis [27].

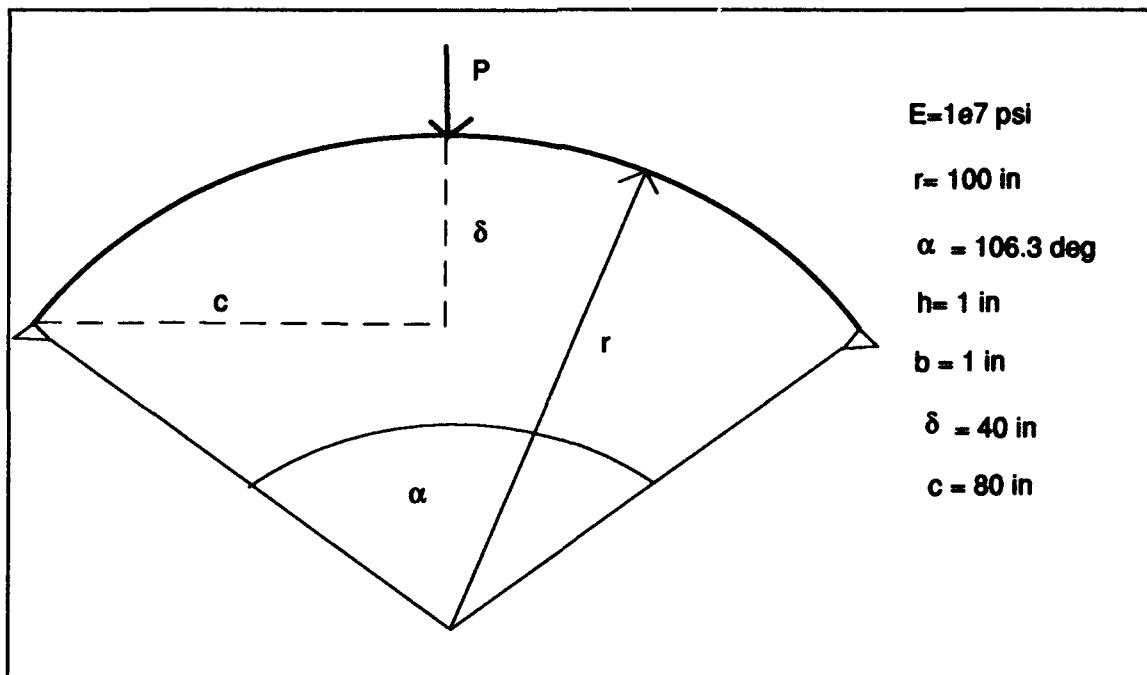


Figure 3-19 Hinged Hinged Deep Thin Arch

Huddelston [17] presents closed form solutions to arches with varying degrees of mid-plane extensibility. He defines the degree of extensibility of an arch mid-plane by a factor:

$$C = \frac{I}{A(2c)^2} \quad (3-6)$$

where I is the moment of inertia, A is the cross sectional area, and c is the dimension shown in figure 3-19. The inextensible solution is when $C=0$. When $C=0.01$, the mid-plane is allowed to stretch or compress. In this case, the arch is initially in compression (until collapse) making the inextensible solution the stiffest. The problem was analyzed with the current theory using a symmetric half arch model with 30 elements and 150 active degrees of freedom. Displacement control and Riks method produced nearly identical results. Figure 3-20 shows the comparisons of a Donnell shell analysis, Creaghan, two Huddelston extensibility solutions, and the current theory.

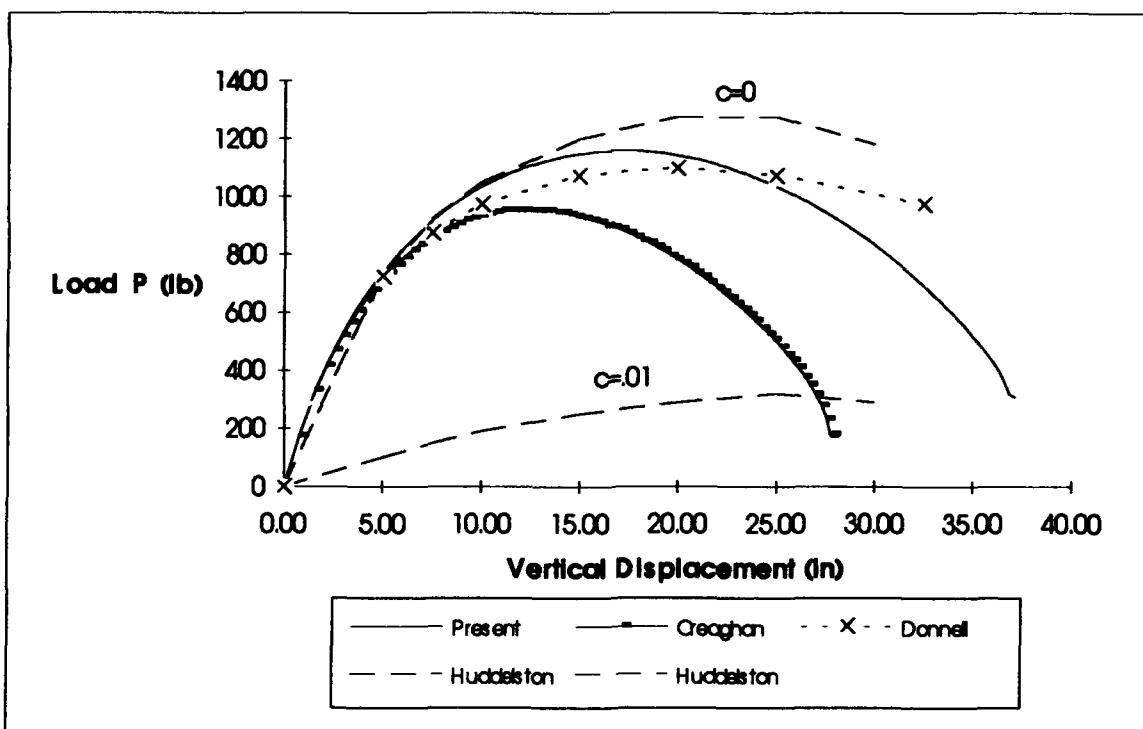


Figure 3-20 Load vs Vertical Displacement for Hinged Hinged Deep Arch

The current solution has produced a much higher collapse load by incorporating the large rotation kinematics. The current theory has also shifted the results toward an inextensible Huddelston solution and a Donnell shell solution. This is an expected result since middle surface extensibility is seen to increase arch deflection under load. The larger rotation theory is allowing more energy to go into cross sectional bending prior to collapse leaving less energy for mid-plane extension. This result becomes more pronounced as rotations get large. This is observed when the current and Creaghan solutions start out close together but diverge as displacements become large. When the displacements do become large, the current theory tends more to an inextensible arch. The peak load has increased by nearly 30% over Creaghan's solution. Donnell shell theory has a much lower order mid-plane strain representation than the current work [27]. The form of the post collapse response for the current work is different than Huddelston and Donnell shell theory, due to the higher order representation of the mid-surface deformation [27]. The deformed shapes of the arch right at collapse and post-collapse are shown in figure 3-21. Local snapping is observed on either side of the load. The load doesn't have any horizontal displacement since the boundary conditions are symmetric.

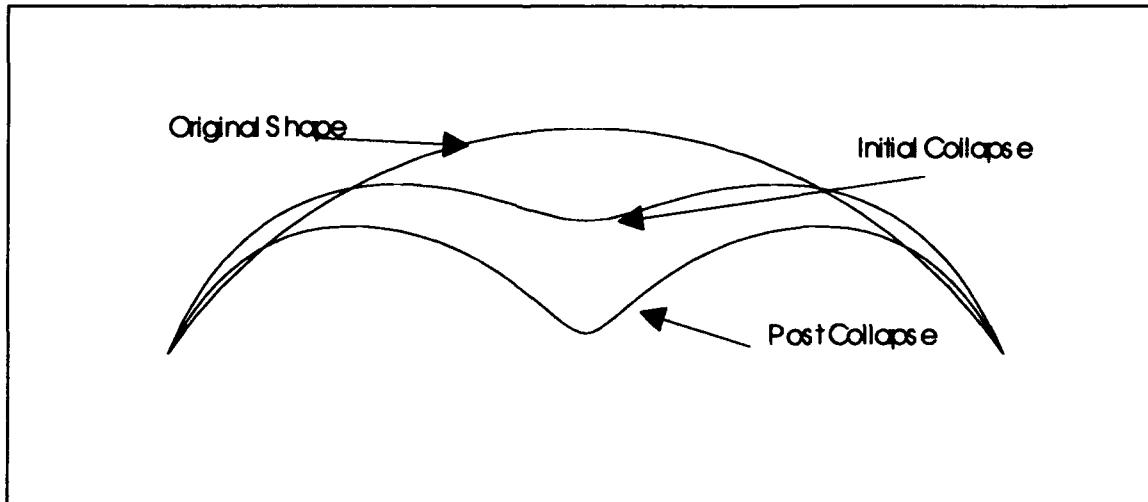


Figure 3-21 Deformed Shapes for Deep Hinged Hinged Arch

Overall, adding large rotation theory has made the response more like an inextensible or a lower order extensibility solution. The bending rotations in this problem reached 46 degrees over 15% of the arch. This large rotation raises the question of shear effects at large rotations. In chapter II, small angle kinematic approximations for bending were used to be consistent with linear shear strain. If instead of making those approximations, the displacement equations of Eqns(2-18) (which include the tangent function) are used to calculate shear strain, bending locking occurs. Figure 3-22 shows the current solution with and without the small angle approximation used in the shear strain relations.

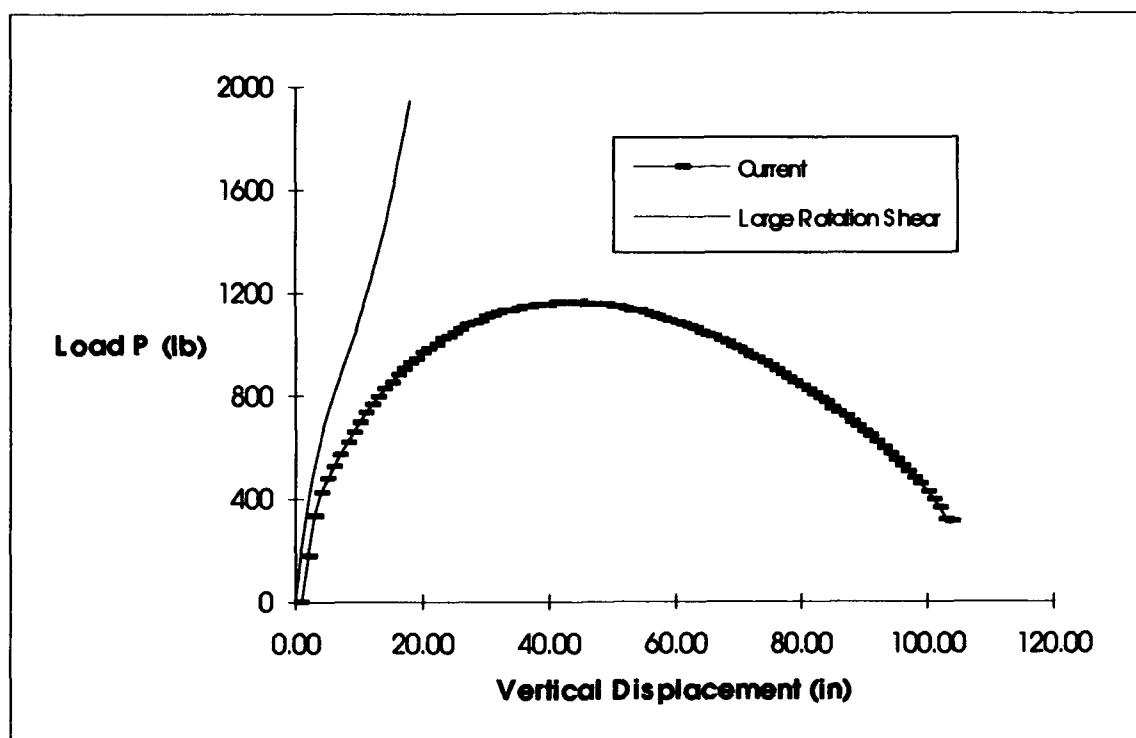


Figure 3-22 Shear Locking Deep Hinged Hinged Arch

If the tangent function (Eqns(2-18)) is included in the shear displacement equation, many of the structures analyzed (not all) exhibit "locking" behavior. For problems where the bending angle is significant (greater than approximately 25 degrees), locking can

occur. A collapse load is never reached in figure 3-22 as the structure becomes very stiff. Using large rotation relationships in the shear equation place displacement gradient values (ψ) into the constant coefficient stiffness matrix K . This makes K a nonlinear function of displacement and causes the locking shown at large rotations.

3.9 Hinged Clamped Isotropic Very Deep Arch #1

The final two problems examined proved to be the biggest challenge for the current theory. The first is a very deep arch with one end pinned and the other end clamped. It is loaded at the crown with a transverse concentrated load. As shown in figure 3-23, this problem has a very large opening angle (240 degrees). The deep arch coupled with the unsymmetrical boundary conditions make this a difficult problem.

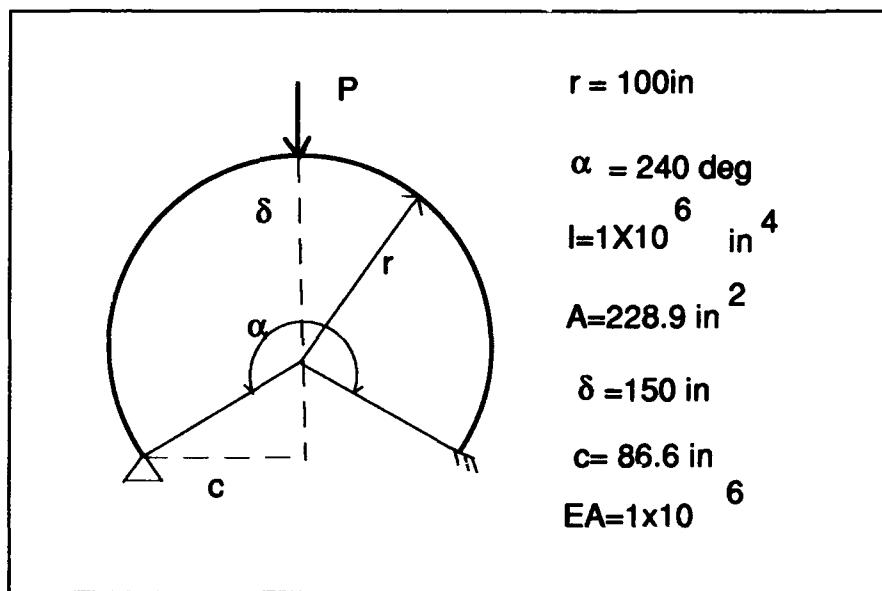


Figure 3-23 Very Deep Hinged Clamped Arch #1

Karamanlidis, Honecker and Knothe (K,H&K) [18] studied this problem using an updated Lagrangian formulation. They incorporate correction terms to an incremental energy function to keep the simulation close to the true solution. The corrections come from stress equilibrium and compatibility. K,H&K include in-plane strain but neglect all other strain components. Since this arch is thin (0.223 in), transverse shear is probably not significant. Only linear strain terms in z are retained and shallow arch assumptions are used for the incremental strain relations. Since they use an incremental approach (assuming the increments are small), shallow approximations are adequate.

This problem was initially analyzed using a full arch model with 40 elements and 198 active degrees of freedom. As the arch length is 418.9 inches, this appears to be a coarse mesh. The results of this analysis are compared with K,H&K in figure 3-24.

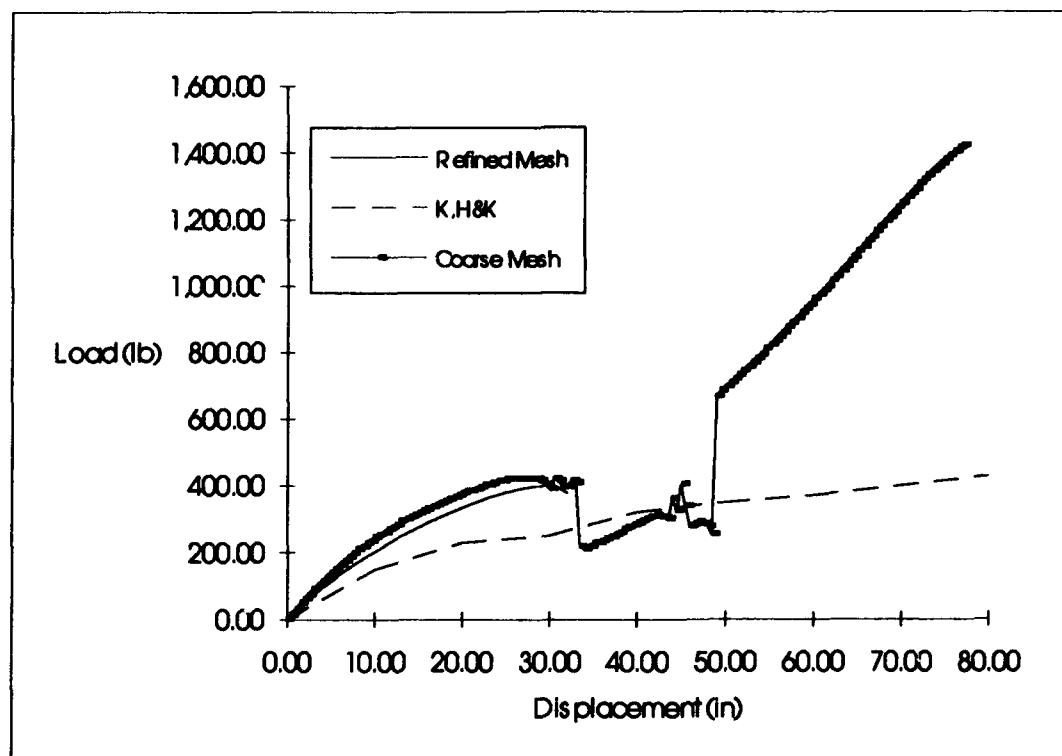


Figure 3-24 Mesh Refinement Very Deep Hinged Clamped Arch #1

The coarse mesh solution becomes unstable near 31 inches of crown vertical displacement. Beyond this point, the results look very questionable. To further examine the response, the deformed shapes of the coarse mesh model are plotted in figure 3-25. When the load starts to drop off, elements near the hinged boundary will "kink" over. They seem to experience a large amount of motion in a single displacement increment. If the structure continues to deform past 31 inches, multiple elements kink until 49 inches. Then the arch once again is able to pick up load, but the boundary condition has changed radically making it a new physical problem. The results in this area of the load displacement curve are not physically correct.

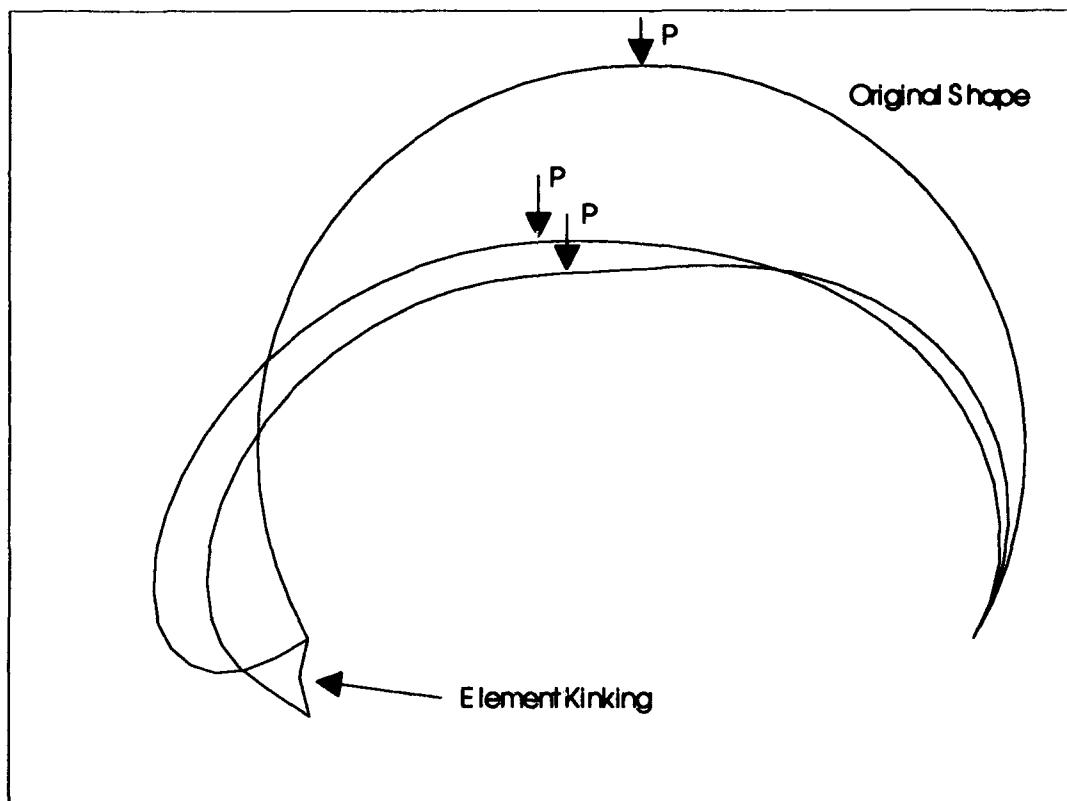


Figure 3-25 Deformed Shapes Showing Element Kinking for Very Deep Hinged Clamped Arch

The deformed shapes of the elements are shown as straight lines, but are actually formed by the element shape functions and the values of the element degrees of freedom at a particular deformation state. Figure 3-25 was produced by drawing straight lines between nodal displacement coordinates at different deformation states. Feeling that the poor results may be due to the coarseness of the mesh, numerous refined meshes were generated. The refined mesh results in figure 3-24 are based on a 170 element model (848 active degrees of freedom) with 130 of the elements placed in the first 40 inches near the pinned boundary. The refined solution follows the same trend as the coarse mesh, but fails to obtain a convergent solution after 31 inches of vertical deflection. No element kinking occurred with the refined mesh and all the deformed shapes remained smooth through out the loading. All of the solutions shown were generated using displacement control. In an attempt to ensure that this limit point wasn't a snap back point that displacement control couldn't capture, various displacement "jumps" were attempted to traverse this point. All attempts to cross this point with a refined mesh failed and resulted in non convergent solutions. Riks technique failed at the same point and actually started to reverse the loading (this is shown in detail in the next problem).

With confidence that the limit point wasn't just a snap back point unable to be captured by displacement control, the code was modified to calculate the internal strain energy in an element using the relationship from chapter II:

$$U = \frac{1}{2} b \int_I d^T \left[\hat{K} + \frac{\hat{N}_1}{3} + \frac{\hat{N}_2}{6} \right] d ds \quad (2-56)$$

The details of how the energy calculations were conducted (including integration) are found in appendix A.

The internal strain energy of elements near the pinned boundary showed a smooth increase until reaching the limit point in figure 3-24. At that point, these elements showed

an instantaneous energy increase by orders of magnitude in many cases. Elements away from the pinned boundary maintained a relatively smooth energy increase throughout the loading. Figure 3-26 shows the strain energy vs vertical displacement of the load for a typical element near the pinned boundary (one that kinked) and an element well removed for the coarse mesh solution. Although they need not have similar values, they should follow the same smooth trend for a properly posed problem.

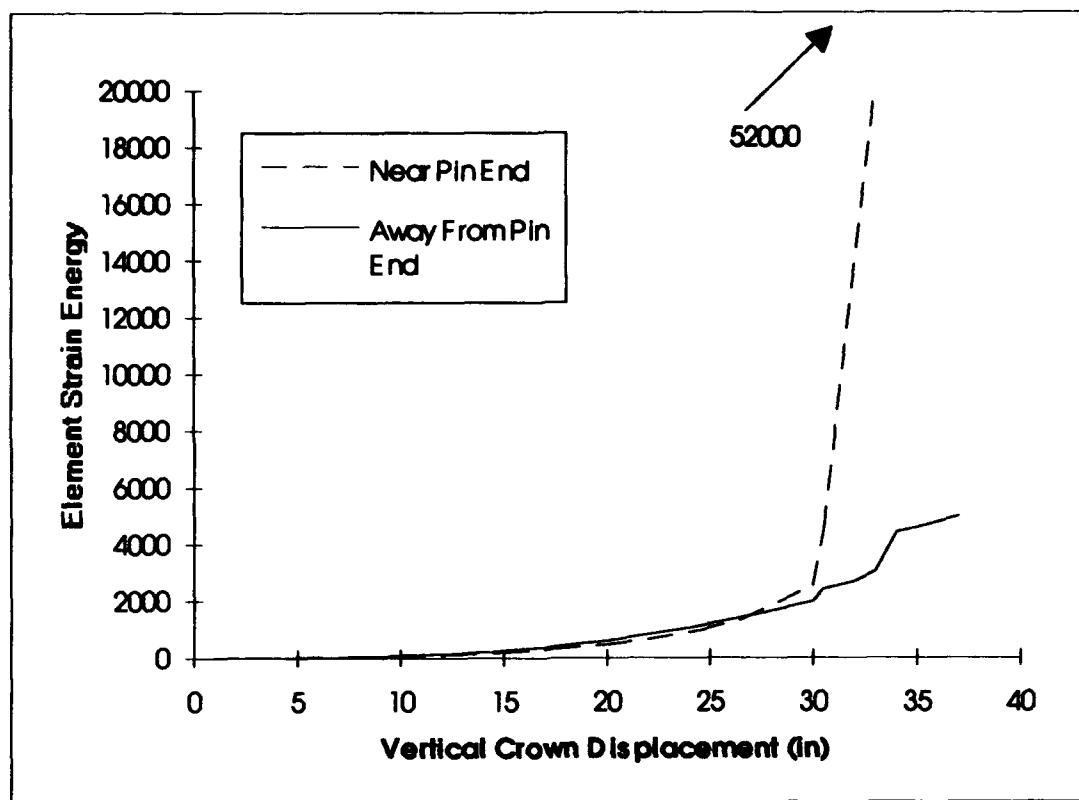


Figure 3-26 Strain Energy for Typical Elements

It is evident that the elements do not follow the same trends and that an extreme increase in the strain energy in a single displacement increment is not physically realistic. This energy jump was experienced in elements near the pinned end for both the coarse and refined mesh models. The structure now contains an unrealistic amount of energy which

points to a numerical problem probably stemming from the element itself. As the energy builds rapidly at one end of the arch, the energy continues to increase elsewhere in the structure at a "normal" rate. There is no energy release to account for the rapid growth at the pinned end, yet the work of the concentrated force is the only energy addition entering the system. Rotations at the pinned end are relatively large at kinking (60 degrees) and are the major contributor to the energy jump. The coarse mesh model converged to a solution (an incorrect one) after the limit point while the refined mesh model didn't. Since a majority of the refined mesh elements are placed in an area of strain energy instability, they occupy a majority of the global stiffness matrix entries. This explains why the refined mesh would not converge to a solution where the coarse mesh did. This implies a numerical problem with the equations that the model develops for the iterative solution. To further confirm this hypothesis, the global stiffness matrix was examined throughout the loading cycle. As the Newton-Ralphson technique tried to find solutions at this limit point (iterating load for a specific displacement), some values on the diagonal of the global stiffness matrix became negative, confirming that the problem is no longer being physically modeled correctly. A negative diagonal stiffness entry implies that a negative displacement takes place from a positive load application which is not possible for a finite element potential energy model. The negative diagonal entries were of the same order of magnitude as other stiffness values. These entries were not isolated to specific types of degrees of freedom, but appeared sporadically in many places. Cook [7] states that any negative or zero stiffness entry on the diagonal implies an unstable structure. As the Newton-Ralphson solver tries to attain increment convergence by iterating, many stiffness entries for elements near the pinned end became much larger than for the other elements. This has the same effect as placing stiff and compliant members next to each other. Cook [7] also shows that very stiff members attached to compliant elements can cause numerical instability.

These energy related problems stem from the large bending rotations experienced at the hinged end due to the unsymmetrical boundary conditions. Each element contains only two bending degrees of freedom, and they are interpolated linearly. This is much too crude for such large angles. The problem could be improved by adding more bending degrees of freedom and using a higher order interpolation function. C^1 shape functions would also help to provide slope continuity to ψ which isn't currently present. For example, at kinking of the coarse mesh, the second element from the pinned end has $\psi_{,\eta} = -0.11207$ and the adjacent element has $\psi_{,\eta} = 0.012455$. The rates of change of the bending angle is discontinuous at their common node. Bending slope continuity would help smooth the bending rotations at the pinned end, resulting in less bending rotations and more stiffness at large rotations. This problem also exceeds assumptions made in the kinematics. Kinematic derivations were based on the assumption that vertical displacement is constant through-the-thickness. This led to using a tangent function which assumed only horizontal motion of cross section points during bending (figure 2-2). As ψ gets large, the in-plane displacement will become infinite. Including normal strain through-the-thickness would help this problem by allowing the cross section to contract making vertical displacement a function of the thickness coordinate. More exact rotation kinematics could then be used in Eqn(2-14) (resulting in a sine function). This would provide more accurate bending coupling characteristics.

3.10 Hinged Clamped Very Deep Isotropic Arch #2

The final problem analyzed is a very deep hinged clamped arch which is similar to the previous problem except that it has a smaller opening angle and thicker cross section. As shown in figure 3-27, this arch is loaded transversely by a concentrated force at the crown. The unsymmetrical boundary conditions will again provide large rotations at the hinged

end. The objective of examining this problem is to compare to other theories and to explore a Riks solution in more detail after displacement control fails to converge. This problem was examined by Creaghan [8], Brockman [5] and DaDeppo and Schmidt [11]. Brockman felt the geometric complexity of this problem required a 3-D analysis. He used an updated Lagrangian code developed for material and geometric nonlinear analysis. To remain consistent with beam assumptions used by others, Brockman used a 3-D element but displacements normal to the plane of bending were suppressed to compare with other theories that didn't account for finite width. DaDeppo and Schmidt conducted an analysis based on the Euler nonlinear theory of elastica. They used a total Lagrangian formulation but had an inextensible mid-plane and no transverse shear. This problem will have more through-the-thickness shear than the previous since it is over four times thicker.

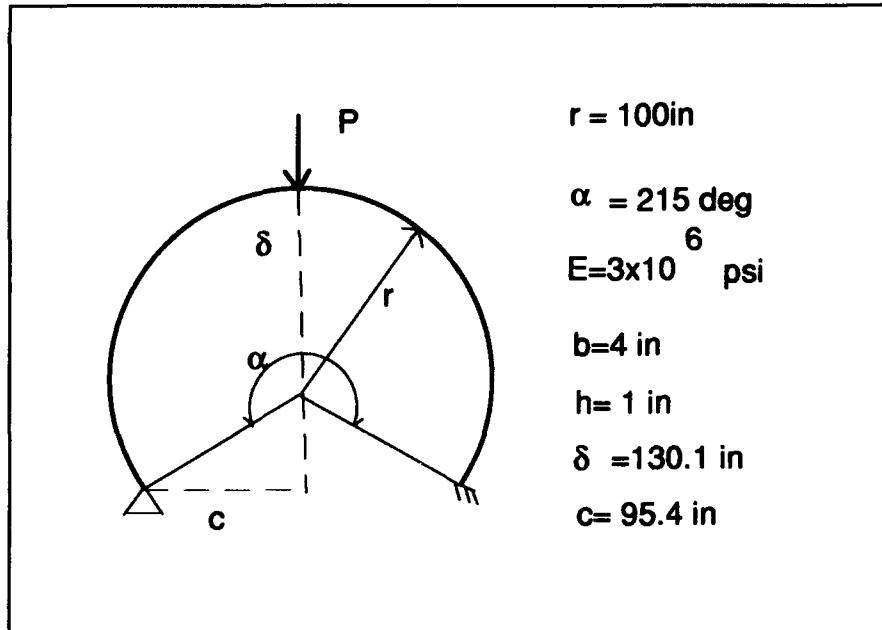
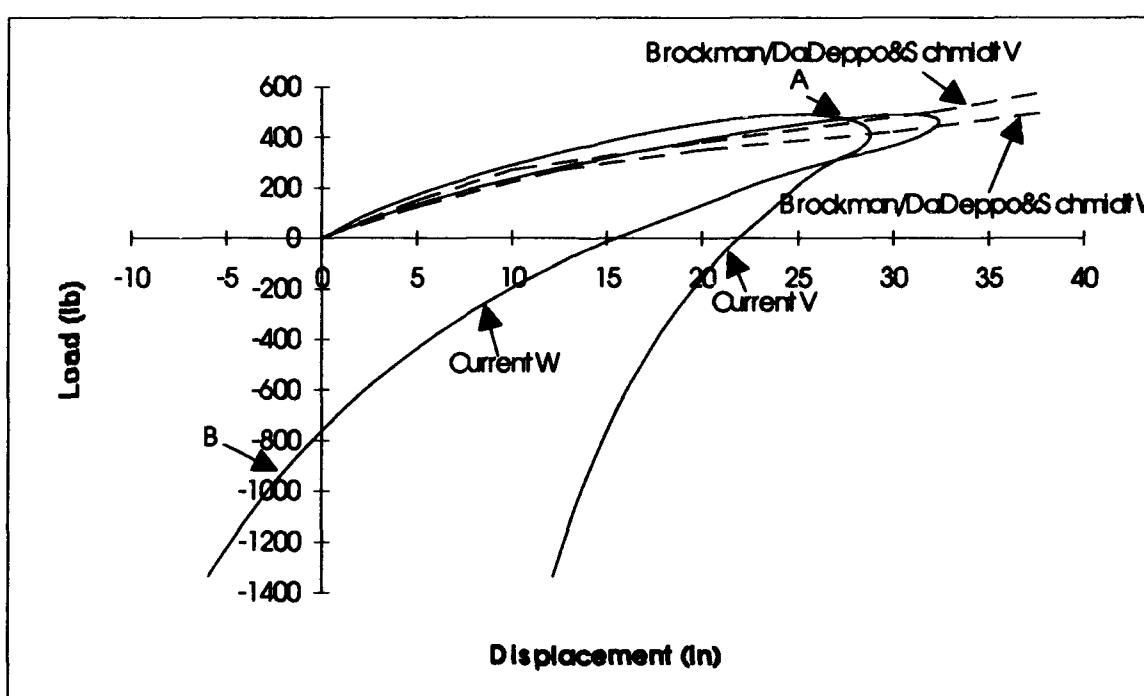
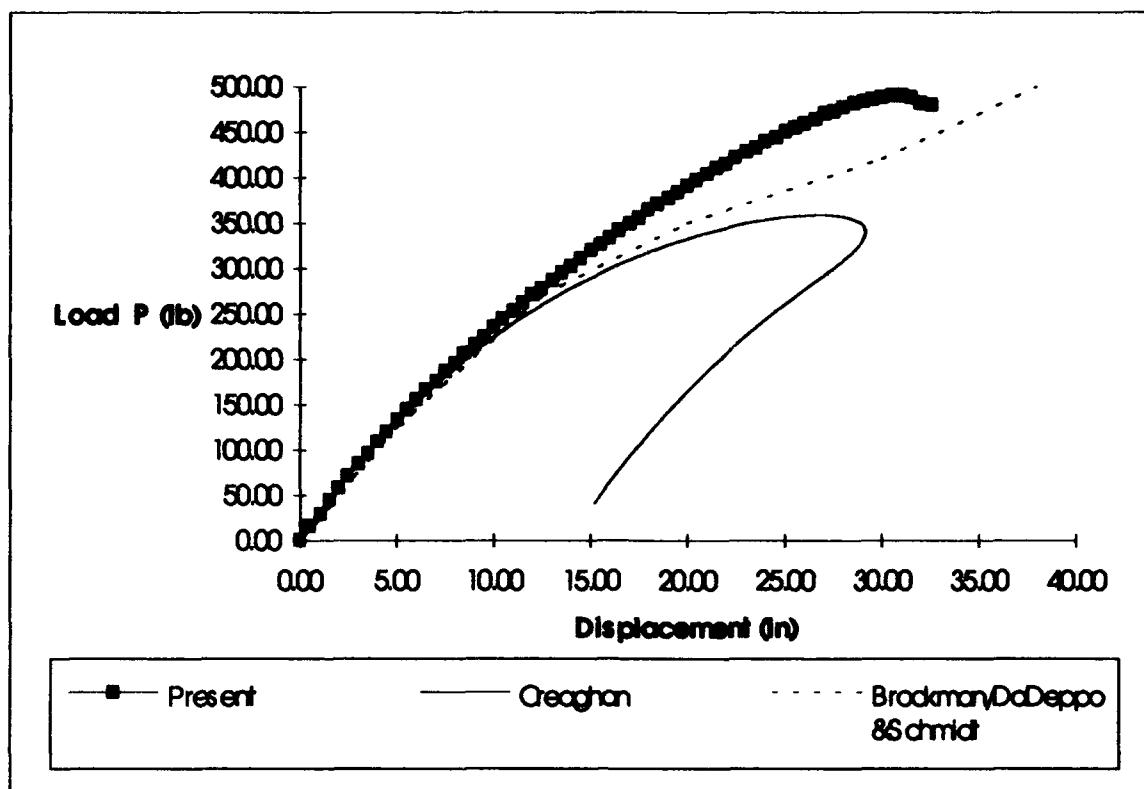


Figure 3-27 Very Deep Hinged Clamped Arch #2

Initially, the analysis was conducted with a 40 element (198 active degrees of freedom) full arch model. The results were very similar to those for a coarse mesh in section 3.9. The elements near the hinged boundary became unstable and kinked. This problem also exhibited the same energy characteristics as the previous one. The mesh was refined to a total of 180 elements (898 active degrees of freedom) with 136 of the elements comprising half of the arch with the pinned end. Displacement control was initially used to obtain equilibrium solutions and help characterize the arch behavior. The results are shown in figure 3-28 with Creaghan and Brockman/DaDeppo and Schmidt.

The overall trend of the current model is similar to the previous problem. There appears to be a limit point near 33 inches of vertical crown displacement (33 times the thickness). Displacement "jumps" were again attempted to ensure that the limit wasn't a snap back that displacement control couldn't capture, but all attempts were non convergent. Creaghan showed solutions within 10% of Brockman/DaDeppo and Schmidt at 17 inches of crown displacement. The current solution starts to diverge sooner (around 15 inches) but attains a higher load and displacement. The current solution gave stable solutions for nearly 5 more inches of displacement and over 150 lb of force than Creaghan was able to attain, although the differences from other solutions should not be neglected. Brockman shows geometric collapse load of 897 lb, while the current solution only reached 491 lb. Brockman also showed rotations near the hinged boundary in excess of 120 degrees when loads approached collapse. The current work is limited to rotations that are far smaller than those experienced by this problem. In an attempt to find other valid equilibrium states for a refined mesh, Riks method was used with the results shown in figure 3-29.



The Riks solution initially follows the same equilibrium path as displacement control until reaching the vertical limit point. As Riks searches along an arch length for equilibrium points, it attempts to get past the limit point, but cannot find a stable state. The first stable state is found when load and displacement both begin to decrease. This is why the curves in figure 3-29 wrap back under. Rotation errors causing strain energy inconsistencies in a majority of the elements don't allow for a physically valid solution beyond this point. Riks technique provides another equilibrium path. The load and displacement in the vertical direction (w) continue to decrease and never turn back. This trend provides an excellent example of how Riks provides other solutions when numerical difficulties are encountered. Looking at the deformed beam shapes gives physical meaning to the strange looking equilibrium solutions generated by Riks. Equilibrium points in figure 3-29 are shown as deformed shapes in figure 3-30. Point A is where the model reaches the maximum positive displacement and positive load. Figure 3-30 shows the deformed shape at this point. As the arch is unloaded by, the deformed shape of a typical point after the load sense has reversed is seen at point B. The magnitude of the upward force and displacement continue to increase. The solution never returns to a positive load (downward) since Riks technique cannot locate other stable equilibrium states in its search radius.

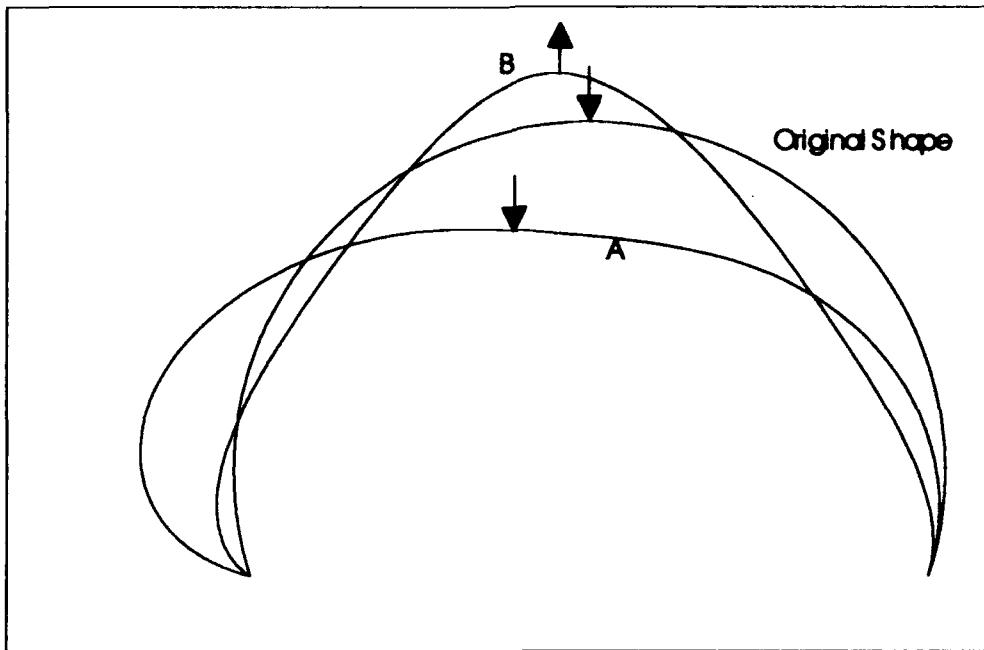


Figure 3-30 Deformed Shapes of Very Deep Hinged Clamped Arch

The changes in bending rotation moving along the arch from the hinged boundary are large. The bending angle changes by nearly 60 degrees from the crown to the hinge when the limit load is reached. With a 1 inch thick cross section, the kinematic assumptions cause difficulty for the model. C^1 shape functions for ψ would help improve the solution by providing slope continuity as the bending angles go through radical changes. At 30 inches of vertical crown deflection, the hinged bending rotations are 40 degrees. At this point, the current load differs from Brockman/DaDeppo and Schmidt by 15%. 40 degrees of bending rotation has been consistently providing accurate solutions for boundaries with unconstrained ψ while problems with ψ constrained at each boundary have given accurate solutions to 45 degrees of bending rotation.

IV Vectorization and Parallelization Considerations

4.1 Vectorization

Advances in computer hardware and software have made more efficient computation possible. Nonlinear finite element programs in particular can involve very large processing times on serial machines [1]. Large degree of freedom systems or conducting dynamic analysis coupled with higher order structural theories can result in extremely large processing times on single processor scalar machines. Multiprocessing technology has emerged as a viable method to possibly decrease processing times and thus increase efficiency. This allows larger dimensional and more complex models to be analyzed. The current effort focused on parallel processing and vector pipelining of tasks.

We begin by examining vector processing. Vector computations refer to several independent data streams being computed on a single processor. Vector computers use hardware processing units called vector pipelines. A vector pipeline allows some operations that are traditionally conducted in a serial manner to take place in parallel and independently on a single pipeline processor. Adeli et al. [1] shows three ways of decreasing the time required to perform specific instructions on a vector processor:

1. Increase the number of vector pipelines in use. An N -fold increase in pipelines can theoretically result in an N fold decrease in processing time when independent tasks are conducted concurrently. This is true to a point since data communication from memory to pipelines and pipeline filling will limit the improvement when the number of pipelines becomes great. The processing improvement is not linear with large processing pipelines because communication time can become significant.

2. Decreasing the precision of the operation can decrease the processing time. On some machines, going from 64 bit operations to 32 bit halves the time requirement. This is possible because a single 64 bit pipeline can be split into two 32 bit pipelines.

Additional pipelines can be used for double precision operations, but if single precision is sufficient, some single pipelines may be split into two pipelines.

3. Special vector operations that involve three input quantities are called linked triadic operations. Frequent scalar and vector operations of these quantities are linked together as one process by extending pipelines to include vector/scalar multiplication and addition. Depending on the specific hardware used, various vector operations may already be resident in the pipelines. When these tasks are conducted concurrently, computation times may be greatly reduced over serial operations [36].

When considering a specific code or set of instructions, there are two methods or levels of vectorization [36]. The first, syntactic vectorization, involves replacing certain lines of code with vector equivalents. This usually requires modifications to an existing code to make it vector compatible. The second level of vectorization is algorithmic. Algorithmic vectorization involves replacing total algorithms with their vector equivalent. This is a much more complete method from a vector standpoint. Algorithmic vectorization is a bottom up type operation as the entire program is modified and designed around vector performance.

VanLuchene, Lee and Meyers [36] state that taking existing computer code and making minor modifications for use on a vector machine (syntactic vectorization) is the most common vectorization technique. They study an updated Lagrangian finite element formulation to evaluate vectorization. An updated Lagrangian is used because of its incremental nature and simpler strain displacement relationships as compared to a total Lagrangian [36]. The current total Lagrangian formulation contains complicated strain displacement relations which may prove to be less advantageous for vectorization. To

investigate the vector possibilities of the model, the code was placed on a Convex 220 vector computer. Our objective is to employ some level of syntactic vectorization to quantify the speed improvement. Since an algorithmic vectorization was not attempted, a knowledge of what areas of the code are the most computationally intensive would help us decide where to concentrate vectorization efforts. Farhat, Wilson and Powell [13] imply that the majority of computational effort for a linear finite element model lies in solving the system of algebraic equations. VanLuchene et al. [36] approximate that 38% of the total CPU time is consumed by forming the element stiffness arrays in a typical nonlinear finite element model. The nonlinear nature of the current model does not make it obvious what part(s) of the code are the most computationally intensive. The current model uses a Gauss elimination technique on the global equations that are stored in a banded symmetric format. VanLuchene et al. [36] discusses a hypermatrix storage scheme of the global stiffness matrix. The advantages seem minimal over the banded symmetric scheme currently used since hypermatrix schemes are elegant (from a matrix algebra perspective) and usually involves more data communication transfers (overhead). Efficient vector Gauss elimination schemes for banded symmetric systems exist and could be used. A study was conducted to determine which routines used the most CPU time in the current model to help focus the vectorization effort. CPU time was recorded for each routine in the code for several typical arch problems. The problems were analyzed on a scalar Sparc 330 computer. The time required for the Gauss elimination scheme of the global system of equations was a surprising 0.6 - 0.7% of the overall CPU time. This was much lower than expected. Most literature for efficient computing of finite element codes shows that the solver usually dominates the CPU time (even for nonlinear problems). 76 - 82% of the CPU time was spent forming element stiffness entries with the remainder of the time in post processing, I/O operations, and other miscellaneous tasks. The element stiffness computations are dominating because \hat{N}_1 and \hat{N}_2 stiffness arrays are recalculated at every

iteration of every increment (whether it is load or displacement control) for each element.

As seen in the code listing in the appendix C, the entries for these arrays involve thousands of algebraic manipulations. After \hat{N}_1 and \hat{N}_2 are formed, they are integrated over the element length. As shown in chapter II, five point Gauss quadrature is used to do the integration. The shape functions of Eqn(2-58) are calculated at each Gauss point for each element. All of this is accomplished in a serial manner and is dominating the computation time. Even for models with a small number of elements, the element stiffness calculations dominate. Now our vectorization effort should achieve the biggest gains by concentrating on the element stiffness evaluation portions of the code. The mathematical operations involved in forming \hat{N}_1 and \hat{N}_2 are very well suited for vectorization because they each involve extensive independent algebraic calculations. The code was slightly modified, then complied using the vectorization (-O2) option on the Convex 220. Approximately 50% of the program loops fully vectorized. The \hat{K} , \hat{N}_1 and \hat{N}_2 routines vectorized completely. As expected, significant reduction in computation time was observed. Typical problems (ranging from 100 to 800 degrees of freedom) were analyzed in scalar mode on a Sparc 330, scalar mode on the Convex and vector mode on the Convex. As shown in figure 4-1, the overall CPU time required for a vector run was 1/8 the time required for a scalar run on the Convex. VanLuchene et al. [36] show that input/output requirements can be reduced by a factor of approximately 100. Input/output time was not monitored in the current study, but similar improvements could be expected.

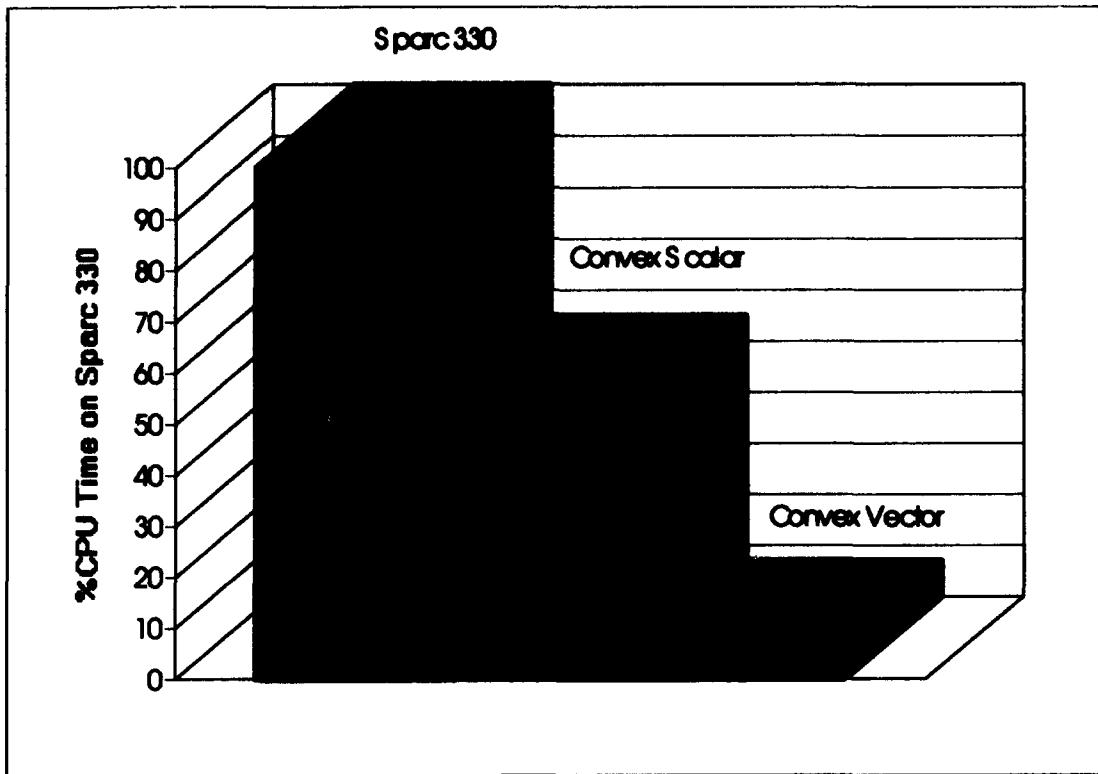


Figure 4-1 CPU Time Comparisons

The results in figure 4-1 confirm the findings of the initial study that the formation of the element stiffness arrays dominate the current model. A CPU time decrease of a factor of 10 is the most that can be expected from a full algorithmically vectorized code over its scalar equivalent [36]. Completing a short syntactic vectorization centered on the element stiffness arrays resulted in a factor of 8 reduction in CPU time. A complete vectorization of the model would gain little more than already shown since a CPU time reduction of a factor of 8 is close to the maximum expected factor of 10 [36].

4.2 Parallel Processors

Another area of advanced computational capability is the use of multiple processors on a single problem. We will look at what others have published in this area on similar

problems and show how concurrent processing could be applied to the current effort in the future. No actual parallel computing of the code was attempted, but knowledge of the algorithms and process allow us to study parallel implementation without actually doing it. Multiple processors can run concurrently on independent tasks to reduce the overall computation time. Parallel operations can bring great increases in computing efficiencies over a single processor. As the number of processors increases, overhead operations tend to get magnified with communication bottlenecks possible. The coding language also has an effect on the parallelization of a program. Adeli et al.[1] prefer FORTRAN 90 for its ease of use in the parallel environment. The current code is in FORTRAN 77 and could be used in a parallel machine (depending upon the specific compiler used), but FORTRAN 90 is better suited for parallel operations and conversion is recommended. The current model has many functional areas that would benefit from parallel processing. As already discussed, formation of the element stiffness arrays during every iteration of every increment for each element consumes the majority of the CPU time for the current model. Each of these operations are independent making them good candidates for a parallel effort. Parallel compatible compilers would identify some of these independent loop operations and parallelize them automatically. Farhat et al. [13] have done extensive work in the area of parallel computing with large finite element models. They take a problem that is already discretized into a finite element system and further subdivide groups of elements. Recognizing that element operations are independent in nearly all finite element models (linear or nonlinear), they assign groups of elements to individual processors. The objective of the subdivision is to produce units that have nearly the same computational load. The domain is usually decomposed into a number of substructures equal to the number of processors available. If the number of processors exceeds the number of elements, then some individual tasks associated with each element can be carried out independently. Each substructure contains the number (rounded in some

cases) of elements assigned to each processor. Figure 4-2 shows how an 8 element arch model may be divided for three processors.

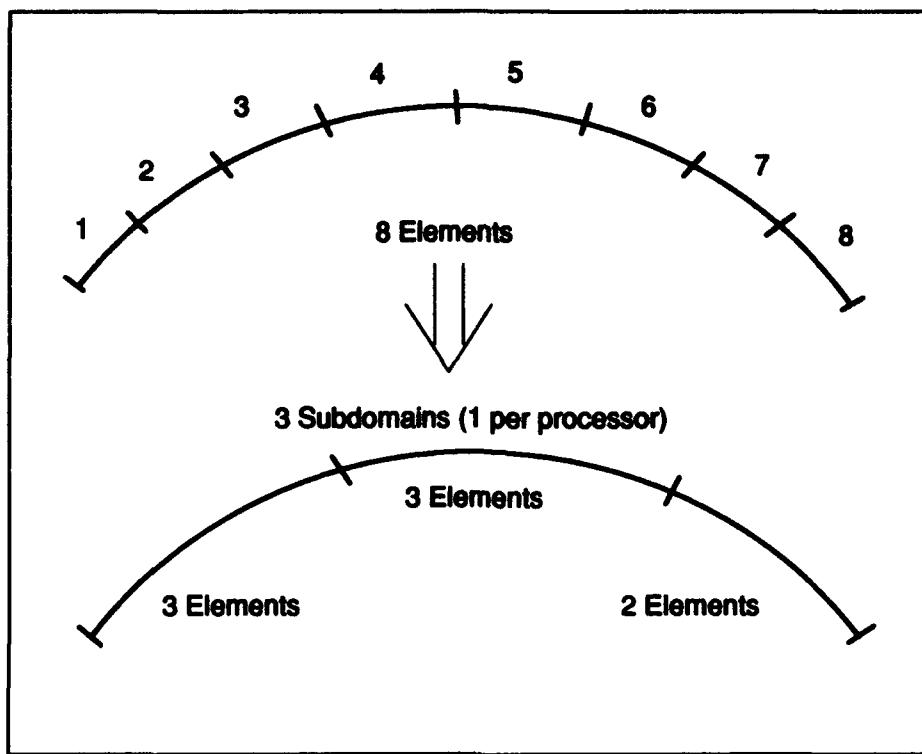


Figure 4-2 Parallel Processor Domain Substructuring

After decomposition of elements into subdomains that are associated with independent processors, element calculations can be conducted in parallel at each iteration of each increment. Figure 4-3 represents an algorithmic flow of the formation of the element stiffness arrays, which have already been shown to be computationally intensive. For serial operations, the element stiffness arrays are generated in order. For a parallel scheme, multiple elements can be formed concurrently. Figure 4-3 also shows how stiffness arrays for a particular element can be formed independently. K is only formed once for a particular element while N_1 and N_2 are updated at every iteration of every

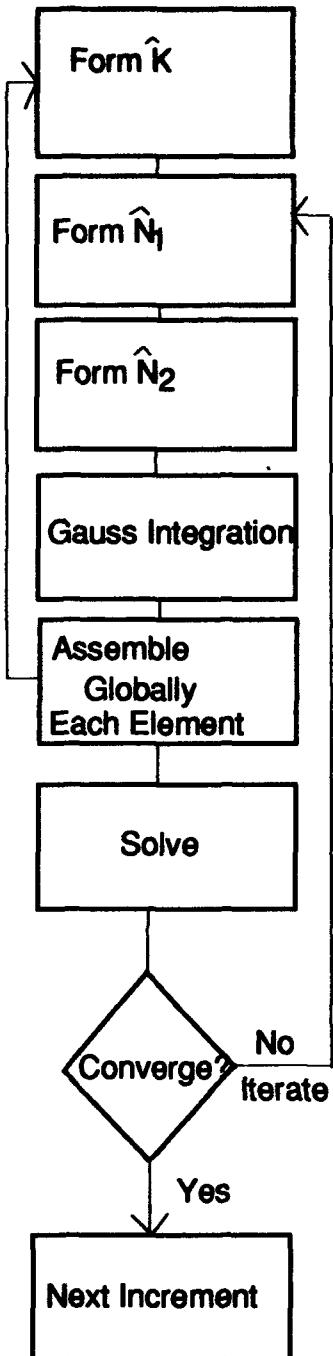
increment. Each requires the same data input, but can be calculated independent of each other. Parallelization provides the possibility of dividing the problem into element subdomains and independent tasks for each element. Now, the number of times the element stiffness calculations are completed on a single processor can go from the number of elements in a model to the number of elements per subdomain.

The concurrent process representation in figure 4-3 has global assembly and system solution steps in serial operations. Farhat, Wilson and Powell [13] have developed a parallel Gauss elimination scheme for this type of problem. If used, the global assembly process is eliminated. The current solver is a serial Gauss elimination scheme that requires global assembly of the equations. If a parallel scheme is used, the solver would reside at each processor and solutions can be obtained without ever assembling the equations globally. Figure 4-3 would be changed so that the system solution is completed in parallel, removing the global assembly block. This parallel solver requires more communication, data transfer and additional arrays to identify global connectivity and sequencing. The current model does not appear to be limited by the solver, but a parallel algorithm is available if desired in the future.

Vectorization and parallelization each provide the current effort the possibility of improving computational performance. With very few changes, vectorization of the time consuming tasks resulted in radically reduced computation times for moderately sized problems. Existing parallel design schemes can be easily adapted to the current effort for use on multiple processors. Dynamic and large degree of freedom systems could benefit greatly from these type of improvements as computation times can easily reach many days on serial machines. Improvements are obtained by breaking the problem down into subdomains and independent element tasks. Many parallel routines already exist and could be used by the current code instead of developing program specific routines. The rate of improvement with concurrent operations will eventually be limited by input/output and

communication operations for a particular architecture. This and many other detailed programming and hardware issues still exist to get the most computational efficiency possible for the current model.

Serial Operations for Each Element



Parallel Operations for Each Subdomain

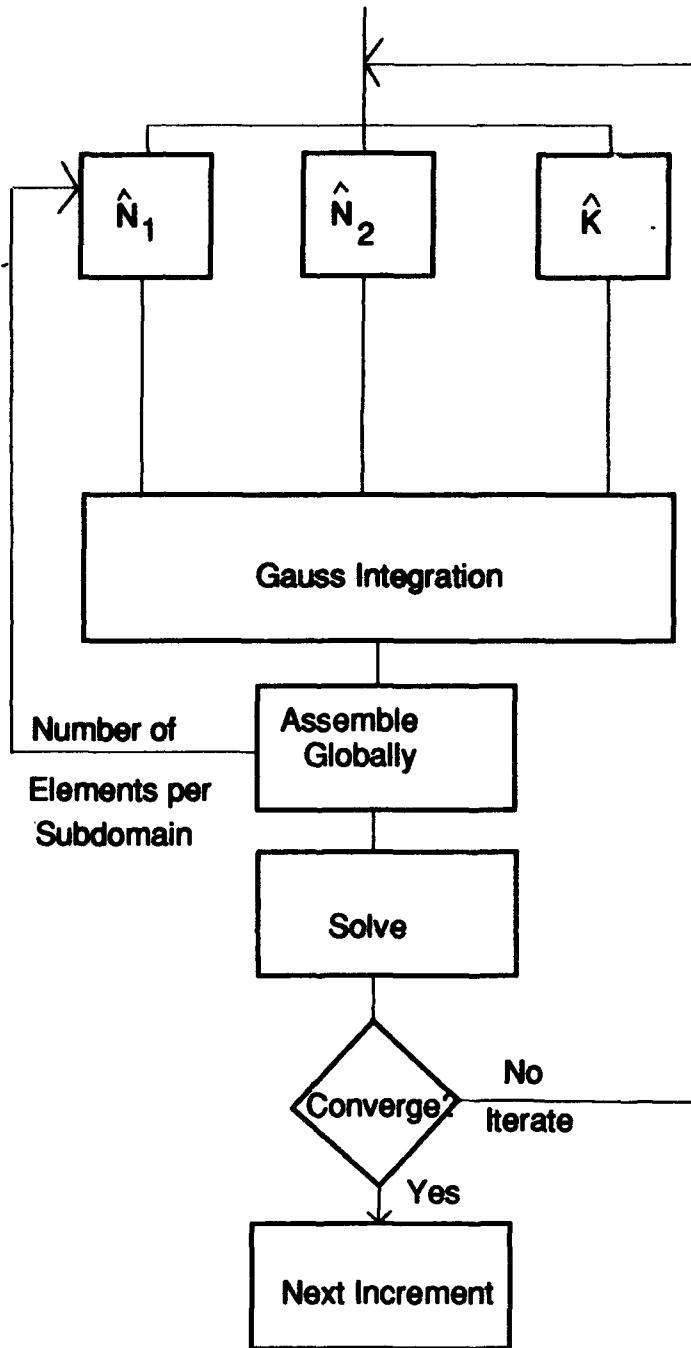


Figure 4-3 Serial and Parallel Element Stiffness Processes

V. Conclusions and Recommendations

5.1 Summary and Conclusions

This effort led to the successful inclusion of a large rotation kinematic theory into a one dimensional geometrically nonlinear arch model. The kinematics were derived vectorally and implemented in the Green strain expressions. A tangent function of the bending angle resulted and was approximated using a truncated series representation. The theory included all the terms for in-plane strain and linear terms for transverse shear strain. All other strain components were neglected. Large rotation kinematics were used only in the in-plane strain equations because of the linear shear assumption. The beam potential energy was derived and the first variation made stationary. This resulted in nonlinear differential equations which were approximated by nonlinear algebraic equations through a finite element scheme. The finite element model was developed by modifying an existing FORTRAN code that was originally based on a two dimensional shell theory. The majority of the code modifications were to the element stiffness arrays. The calculations for each element increased by approximately 75% for the large rotation version of the code.

Numerous problems were analyzed to determine the adequacy of the proposed theory. The first group of problems involved large displacements, but relatively small bending rotations (around 10 degrees). Results from using the previous moderate rotation theory were nearly identical. This was expected since small angle approximations used in moderate rotation theory were accurate. These initial problems also showed that the new kinematics didn't corrupt the model and that the theory was implemented in the code correctly.

Computational efficiency was briefly examined. A syntactic vectorization centered around the formation of the element stiffness matrices resulted in a vector processing time that was 1/8 the time required in scalar mode. Parallelization was examined and possible parallel options discussed.

5.2 Recommendations for Further Work

As large rotations are limiting the current model to 45 degrees of bending rotation, some simple changes to the code could help achieve large rotations and more nonlinearity. Increasing the rotation degrees of freedom per element would allow higher order shape functions to more accurately represent the actual bending angle distribution. Since the slope of the bending rotation is discontinuous at some nodes, Hermitian shape functions for ψ would give consistent rates of change between elements. Incorporating C^1 shape functions would require that a ψ_2 degree of freedom be added to each end of the element. This would smooth the bending rotations and provide more bending stiffness at large angles. If transverse normal strain is added to the formulation, more physically correct kinematics can be derived. If the cross section is allowed to contract or extend, then a sine function is used in place of the tangent for displacement of points undergoing pure bending.

In conclusion, the theory presented has shown significant improvements over the previous moderate rotation theory. Using large angle kinematics resulted in a stiffer structure and more accurately modeled actual beams and arches.

Appendix A

A.1 Q_{ij} Transformations of Eqn(2-21)

$$\begin{aligned}
 \bar{Q}_{11} &= Q_{11} \cos^4 \theta + 2(Q_{12} + 2Q_{66}) \sin^2 \theta \cos^2 \theta + \sin^4 \theta \\
 \bar{Q}_{12} &= (Q_{11} + Q_{22} - 4Q_{66}) \sin^2 \theta \cos^2 \theta + Q_{12} (\sin^4 \theta + \cos^4 \theta) \\
 \bar{Q}_{22} &= Q_{11} \sin^4 \theta + 2(Q_{12} + 2Q_{66}) \sin^2 \theta \cos^2 \theta + Q_{22} \cos^4 \theta \\
 \bar{Q}_{16} &= (Q_{11} - Q_{12} - 2Q_{66}) \sin \theta \cos^3 \theta + (Q_{12} - Q_{22} + 2Q_{66}) \sin^3 \theta \cos \theta \\
 \bar{Q}_{26} &= (Q_{11} - Q_{12} - 2Q_{66}) \sin^3 \theta \cos \theta + (Q_{12} - Q_{22} + 2Q_{66}) \sin \theta \cos^3 \theta \\
 \bar{Q}_{66} &= (Q_{11} + Q_{22} - 2Q_{12} - 2Q_{66}) \sin^2 \theta \cos^2 \theta + Q_{66} (\sin^4 \theta + \cos^4 \theta) \\
 \bar{Q}_{44} &= Q_{44} \cos^2 \theta + Q_{55} \sin^2 \theta \\
 \bar{Q}_{45} &= (Q_{44} - Q_{55}) \cos \theta \sin \theta \\
 \bar{Q}_{55} &= Q_{55} \cos^2 \theta + Q_{44} \sin^2 \theta
 \end{aligned}$$

A.2 L , S and H Matrices in Eqns (2-52) and (2-54)

$$\begin{aligned}
 L_0^T &= \{0 \ 1 \ -c \ 0 \ 0 \ 0 \ 0 \ 0\} \\
 L_1^T &= \{0 \ 0 \ -c^2 \ 0 \ 0 \ 0 \ 1\} \\
 L_2^T &= \{0 \ -c^2 \ 0 \ 0 \ 0 \ 0 \ 0 \ c\} \\
 L_3^T &= \{0 \ 0 \ 0 \ 0 \ k \ 0 \ k\} \\
 L_4^T &= \{0 \ 0 \ 0 \ 0 \ ck \ 0 \ ck\} \\
 L_5^T &= \{0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0\} \\
 L_6^T &= \{0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0\} \\
 L_7^T &= \{0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0\} \\
 S_1^T &= \{0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0\} \\
 S_2^T &= \{0 \ 0 \ 0 \ 3k \ 0 \ 3k \ 0\}
 \end{aligned}$$

$$H_0 = \begin{bmatrix} c^2 & 0 & 0 & c & 0 & 0 & 0 \\ 0 & 1 & -c & 0 & 0 & 0 & 0 \\ 0 & -c & c^2 & 0 & 0 & 0 & 0 \\ c & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} 0 & 0 & 0 & c^2 & 0 & c^2 & 0 \\ 0 & 0 & -c^2 & 0 & 0 & 0 & 1+\psi^2 \\ 0 & -c^2 & 2c^3 & 0 & 0 & 0 & -c(1+\psi^2) \\ c^2 & 0 & 0 & 2c & 0 & c & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c^2 & 0 & 0 & c & 0 & 0 & \psi \\ 0 & 1+\psi^2 & -c(1+\psi^2) & 0 & 0 & \psi & 0 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} -3c^4 & 0 & 0 & -2c^3 & 0 & c^3 & 0 \\ 0 & -3c^2 & 2c^3 & 0 & 0 & 0 & c(1+\psi^2) \\ 0 & 2c^3 & 0 & 0 & 0 & 0 & -2c^2 \\ -2c^3 & 0 & 0 & 0 & 0 & 2c^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ c^3 & 0 & 0 & 2c^2 & 0 & c^2 & c\psi \\ 0 & c(1+\psi^2) & -2c^2 & 0 & 0 & c\psi & 1+2\psi^2 \end{bmatrix}$$

$$H_3 = \begin{bmatrix} 2c^5 & 0 & 0 & kc^2 & 0 & -2c^4 + kc^2 & 0 \\ 0 & 2c^3 & 0 & 0 & k & 0 & -2c^2 + k \\ 0 & 0 & 0 & 0 & -kc & 0 & -kc \\ kc^2 & 0 & 0 & 2kc & 0 & kc & 0 \\ 0 & k & -kc & 0 & 0 & 0 & 0 \\ -2c^4 + kc^2 & 0 & 0 & kc & 0 & 2c^3 & 0 \\ 0 & -2c^2 + k & -kc & 0 & 0 & 0 & 2c \end{bmatrix}$$

$$H_4 = \begin{bmatrix} 0 & 0 & 0 & kc^3 & 0 & kc^3 & 0 \\ 0 & 0 & 0 & 0 & kc & 0 & kc \\ 0 & 0 & 0 & 0 & -2kc^2 & 0 & -2kc^2 \\ kc^3 & 0 & 0 & 4kc^2 & 0 & 3kc^2 & 0 \\ 0 & kc & -2kc^2 & 0 & 0 & 0 & k(1+\psi^2) \\ kc^3 & 0 & 0 & 3kc^2 & 0 & 2kc^2 & 0 \\ 0 & kc & -2kc^2 & 0 & k(1+\psi^2) & 0 & 2k(1+\psi^2) \end{bmatrix}$$

$$H_5 = \begin{bmatrix} 0 & 0 & 0 & -2kc^4 & 0 & -2kc^4 & 0 \\ 0 & 0 & 0 & 0 & -2kc^2 & 0 & -2kc^2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -2kc^4 & 0 & 0 & 0 & 0 & 2kc^3 & 0 \\ 0 & -2kc^2 & 0 & 0 & 0 & 0 & 2kc(1+\psi^2) \\ -2kc^4 & 0 & 0 & 2kc^3 & 0 & 4kc^3 & 0 \\ 0 & -2kc^2 & 0 & 0 & 2kc(1+\psi^2) & 0 & 4kc \end{bmatrix}$$

$$H_6 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & k^2c^2 & 0 & k^2c^2 & 0 \\ 0 & 0 & 0 & 0 & k^2 & 0 & k^2 \\ 0 & 0 & 0 & k^2c^2 & 0 & k^2c^2 & 0 \\ 0 & 0 & 0 & 0 & k^2 & 0 & k^2 \end{bmatrix}$$

$$H_7 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2k^2c^3 & 0 & 2k^2c^3 & 0 \\ 0 & 0 & 0 & 0 & 2k^2c & 0 & 2k^2c \\ 0 & 0 & 0 & 2k^2c^3 & 0 & 2k^2c^3 & 0 \\ 0 & 0 & 0 & 0 & 2k^2c & 0 & 2k^2c \end{bmatrix}$$

A.3 Element Stiffness Matrices of Eqn(2-56)

$$\begin{aligned}
\hat{K} = & AL_0L_0^T + D(L_0L_2^T + L_2L_0^T + L_1L_1^T) + \\
& F(L_0L_4^T + L_4L_0^T + L_1L_3^T + L_3L_1^T + L_2L_2^T) + \\
& H(L_2L_4^T + L_4L_2^T + L_3L_3^T) + JL_4L_4^T + AS S_0S_0^T + \\
& DS(S_0S_2^T + S_2S_0^T) + FS S_2S_2^T
\end{aligned}$$

$$\begin{aligned}
\hat{N}_1 = & A(L_0d^TH_0 + d^TL_0H_0 + H_0dL_0^T) + \\
& D(L_0d^TH_2 + L_2d^TH_0 + L_1d^TH_1 + d^TL_0H_2 + d^TL_2H_0 + d^TL_1H_1 + H_2dL_0^T + H_0dL_2^T + H_1dL_1^T) + \\
& F(L_0d^TH_4 + L_4d^TH_0 + L_1d^TH_3 + L_3d^TH_1 + L_2d^TH_2 + d^TL_0H_4 + \\
& d^TL_4H_0 + d^TL_1H_3 + d^TL_3H_1 + d^TL_2H_2 + H_4dL_0^T + H_0dL_4^T + \\
& H_3dL_1^T + H_1dL_3^T + H_2dL_2^T) + \\
& H(L_0d^TH_6 + L_1d^TH_5 + L_2d^TH_4 + L_3d^TH_3 + L_4d^TH_2 + d^TL_0H_6 + \\
& d^TL_1H_5 + d^TL_2H_4 + d^TL_3H_3 + d^TL_4H_2 + H_6dL_0^T + H_5dL_1^T + \\
& H_4dL_2^T + H_3dL_3^T + H_2dL_4^T) + \\
& J(L_1d^TH_7 + L_2d^TH_6 + L_3d^TH_5 + L_4d^TH_4 + d^TL_1H_7 + d^TL_2H_6 + \\
& d^TL_3H_5 + d^TL_4H_4 + H_7dL_1^T + H_6dL_2^T + H_5dL_3^T + H_4dL_4^T) + \\
& L(L_3d^TH_7 + L_4d^TH_6 + d^TL_3H_3 + d^TL_4H_4 + H_7dL_3^T + H_4dL_4^T)
\end{aligned}$$

$$\begin{aligned}
\hat{N}_2 = & A(H_0 dd^T H_0 + \frac{1}{2} d^T H_0 dH_0) + \\
& D(\frac{1}{2} H_0 dd^T H_2 + \frac{1}{2} H_2 dd^T H_0 H_1 dd^T H_1 + \frac{1}{4} d^T H_0 dH_2 + \frac{1}{4} d^T H_2 dH_0 + \frac{1}{2} d^T H_1 dH_1) + \\
& F(\frac{1}{2} H_0 dd^T H_4 + \frac{1}{2} H_4 dd^T H_0 + H_1 dd^T H_3 + H_3 dd^T H_1 + H_2 dd^T H_2 + \\
& \frac{1}{4} d^T H_0 dH_4 + \frac{1}{4} d^T H_4 dH_0 + \frac{1}{2} d^T H_1 dH_3 + \frac{1}{2} d^T H_3 dH_1 + \frac{1}{2} d^T H_2 dH_2) + \\
& H(\frac{1}{2} H_0 dd^T H_6 + \frac{1}{2} H_6 dd^T H_0 + H_1 dd^T H_5 + H_5 dd^T H_1 + H_2 dd^T H_4 + \\
& H_4 dd^T H_2 + H_3 dd^T H_3 + \frac{1}{4} d^T H_0 dH_6 + \frac{1}{4} d^T H_6 dH_0 + \frac{1}{2} d^T H_1 dH_5 + \\
& \frac{1}{2} d^T H_5 dH_1 + \frac{1}{2} d^T H_2 dH_4 + \frac{1}{2} d^T H_4 dH_2 + \frac{1}{2} d^T H_3 dH_3) + \\
& J(H_1 dd^T H_7 + H_7 dd^T H_1 + H_2 dd^T H_6 + H_6 dd^T H_2 + H_3 dd^T H_5 + \\
& H_5 dd^T H_3 + H_4 dd^T H_4 + \frac{1}{2} d^T H_1 dH_7 + \frac{1}{2} d^T H_7 dH_1 + \frac{1}{2} d^T H_2 dH_6 + \\
& \frac{1}{2} d^T H_6 dH_2 + \frac{1}{2} d^T H_3 dH_5 + \frac{1}{2} d^T H_5 dH_3 + \frac{1}{2} d^T H_4 dH_4) + \\
& L(H_3 dd^T H_7 + H_7 dd^T H_3 + H_4 dd^T H_6 + H_6 dd^T H_4 + H_5 dd^T H_5 + \\
& \frac{1}{2} d^T H_3 dH_7 + \frac{1}{2} d^T H_7 dH_3 + \frac{1}{2} d^T H_4 dH_6 + \frac{1}{2} d^T H_6 dH_4 + \frac{1}{2} d^T H_5 dH_5) + \\
& R(H_5 dd^T H_7 + H_7 dd^T H_5 + H_6 dd^T H_6 + \frac{1}{2} d^T H_5 dH_7 + \frac{1}{2} d^T H_7 dH_5 + \frac{1}{2} d^T H_6 dH_6) + \\
& T(H_7 dd^T H_7 + \frac{1}{2} d^T H_7 dH_7)
\end{aligned}$$

A.4 Element Strain Energy Calculation

Strain energy in selected elements was calculated using Eqn (2 - 56):

$$U = \frac{1}{2} b \int d^T \left[\hat{K} + \frac{\hat{N}_1}{3} + \frac{\hat{N}_2}{6} \right] d ds$$

which can be expressed in natural coordinates as:

$$U = \frac{1}{2} b \int_{-1}^1 d^T \left[\hat{K} + \frac{\hat{N}_1}{3} + \frac{\hat{N}_2}{6} \right] d \operatorname{Det} J d\eta$$

$\operatorname{Det} J$ is the determinant of the Jacobian matrix (half element here)

The integral above can be approximated using Gauss quadrature:

$$U \approx \sum_{i=1}^m w_i \phi(\eta_i)$$

where m is the number of Gauss points (order of quadrature)

w_i is Gauss weight factor for each Gauss point and

$$\phi(\eta_i) = \frac{1}{2} b d^T \left[\hat{K} + \frac{\hat{N}_1}{3} + \frac{\hat{N}_2}{6} \right] d \operatorname{Det} J \quad \text{evaluated at each Gauss point}$$

d is found at each Gauss point by evaluating the shape functions at that point and multiplying by the element degrees of freedom .

\hat{K} , \hat{N}_1 , and \hat{N}_2 are then evaluated with d for each Gauss point

Appendix B. MACSYMA Inputs to Generate N_1 and N_2 Subroutines

B.1 N_1 Inputs

```
WRITEFILE("BEAMN1.WF");

/*********************************************************************
/* MACSYMA ROUTINE FOR ELEMENTAL CODE GENERATION BY S. A. SCHIMMELS */
/* MODIFIED BY CAPT DAN MILLER FOR 1-D BEAMS */
/* CREATED AS A PART OF AN AIR FORCE INSTITUTE OF TECHNOLOGY (AFIT) */
/* PROGRAM IN AERONAUTICAL ENGINEERING -- MARCH 1993 */
/* MACSYMA IS A REGISTERED TRADEMARK OF */
/* THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY */
/* FOR A CURVED BEAM. CREATES ELEMENT */
/* INDEPENDENT STIFFNESS ARRAYS N1 & N1S.
/*********************************************************************
/* INITIALIZE MACSYMA PARAMETERS AND DECLARE VARIABLE PROPERTIES */
/*********************************************************************$[DYNAMALLOC:TRUE,DISKGC:TRUE,DERIVABBREV:TRUE,POWERDISP:TRUE]$DECLARE([K,C,SI],CONSTANT);

/*********************************************************************
/* GENERATE THE NONLINEAR ELEMENT-INDEPENDENT STIFFNESS ARRAY N1.
/*********************************************************************$[Q: MATRIX([Q(1),Q(2),Q(3),Q(4),Q(5),Q(6),Q(7),Q(8))];$Q: TRANSPOSE(Q);$L0T: EMATRIX(3,8,1,2,2)+EMATRIX(3,8,-C,2,3);$
```

```

L1T:EMATRIX(3,8,-C^2,2,3)+EMATRIX(3,8,1,2,7);
L2T:EMATRIX(3,8,-C^2,2,2)+EMATRIX(3,8,C,2,7);
L3T:EMATRIX(3,8,K,2,5)+EMATRIX(3,8,K,2,7);
L4T:EMATRIX(3,8,C*K,2,5)+EMATRIX(3,8,C*K,2,7);
L5T:ZEROMATRIX(3,8);
L6T:ZEROMATRIX(3,8);
L7T:ZEROMATRIX(3,8);

L0:TRANSPOSE(L0T);
L1:TRANSPOSE(L1T);
L2:TRANSPOSE(L2T);
L3:TRANSPOSE(L3T);
L4:TRANSPOSE(L4T);
L5:TRANSPOSE(L5T);
L6:TRANSPOSE(L6T);
L7:TRANSPOSE(L7T);
H0:EMATRIX(8,24,C^2,1,9)+EMATRIX(8,24,C,1,12)+  

    EMATRIX(8,24,1,2,10)+EMATRIX(8,24,-C,2,11)+  

    EMATRIX(8,24,-C,3,10)+EMATRIX(8,24,C^2,3,11)+  

    EMATRIX(8,24,C,4,9)+EMATRIX(8,24,1,4,12);
H1:EMATRIX(8,24,C^2,1,12)+EMATRIX(8,24,C^2,1,14)+  

    EMATRIX(8,24,-C^2,2,11)+EMATRIX(8,24,1+SI^2,2,15)+  

    EMATRIX(8,24,-C^2,3,10)+EMATRIX(8,24,2*C^3,3,11)+  

    EMATRIX(8,24,-C-C*SI^2,3,15)+  

    EMATRIX(8,24,C^2,4,9)+EMATRIX(8,24,2*C,4,12)+  

    EMATRIX(8,24,C,4,14)+EMATRIX(8,24,C^2,6,9)+  

    EMATRIX(8,24,C,6,12)+  

    EMATRIX(8,24,SI,6,15)+EMATRIX(8,24,1+SI^2,7,10)+  

    EMATRIX(8,24,-C-C*SI^2,7,11)+EMATRIX(8,24,SI,7,14);
H3:EMATRIX(8,24,2*C^5,1,9)+EMATRIX(8,24,K*C^2,1,12)+  

    EMATRIX(8,24,-2*C^4+K*C^2,1,14)+EMATRIX(8,24,2*C^3,2,10)+  

    EMATRIX(8,24,K,2,13)+EMATRIX(8,24,-2*C^2+K,2,15)+  

    EMATRIX(8,24,-K*C,3,13)+EMATRIX(8,24,-K*C,3,15)+  

    EMATRIX(8,24,K*C^2,4,9)+EMATRIX(8,24,2*K*C,4,12)+  

    EMATRIX(8,24,K*C,4,14)+EMATRIX(8,24,K,5,10)+  

    EMATRIX(8,24,-K*C,5,11)+EMATRIX(8,24,-2*C^4+K*C^2,6,9)+  

    EMATRIX(8,24,K*C,6,12)+EMATRIX(8,24,2*C^3,6,14)+  

    EMATRIX(8,24,-2*C^2+K,7,10)+EMATRIX(8,24,-K*C,7,11)+  

    EMATRIX(8,24,2*C,7,15);
H4:EMATRIX(8,24,K*C^3,1,12)+EMATRIX(8,24,K*C^3,1,14)+  

    EMATRIX(8,24,K*C,2,13)+EMATRIX(8,24,K*C,2,15)+  

    EMATRIX(8,24,-2*K*C^2,3,13)+EMATRIX(8,24,-2*K*C^2,3,15)+  

    EMATRIX(8,24,K*C^3,4,9)+EMATRIX(8,24,4*K*C^2,4,12)+  

    EMATRIX(8,24,3*K*C^2,4,14)+EMATRIX(8,24,K*C,5,10)+  

    EMATRIX(8,24,-2*K*C^2,5,11)+EMATRIX(8,24,K+K*SI^2,5,15)+  

    EMATRIX(8,24,K*C^3,6,9)+EMATRIX(8,24,3*K*C^2,6,12)+  

    EMATRIX(8,24,2*K*C^2,6,14)+EMATRIX(8,24,K*C,7,10)+  

    EMATRIX(8,24,-2*K*C^2,7,11)+EMATRIX(8,24,K+K*SI^2,7,13)+  

    EMATRIX(8,24,2*K+2*K*SI^2,7,15);
H5:EMATRIX(8,24,-2*K*C^4,1,12)+EMATRIX(8,24,-2*K*C^4,1,14)+  

    EMATRIX(8,24,-2*K*C^2,2,13)+EMATRIX(8,24,-2*K*C^2,2,15)+  

    EMATRIX(8,24,-2*K*C^4,4,9)+EMATRIX(8,24,2*K*C^3,4,14)+

```

```

EMATRIX(8,24,-2*K*C^2,5,10)+  

EMATRIX(8,24,2*K*C+2*K*C*SI^2,5,15)+  

EMATRIX(8,24,-2*K*C^4,6,9)+EMATRIX(8,24,2*K*C^3,6,12)+  

EMATRIX(8,24,4*K*C^3,6,14)+EMATRIX(8,24,-2*K*C^2,7,10)+  

EMATRIX(8,24,2*K*C+2*K*C*SI^2,7,13)+  

EMATRIX(8,24,4*K*C,7,15);  

H6:EMATRIX(8,24,K^2*C^2,4,12)+EMATRIX(8,24,K^2*C^2,4,14)+  

EMATRIX(8,24,K^2,5,13)+EMATRIX(8,24,K^2,5,15)+  

EMATRIX(8,24,K^2*C^2,6,12)+EMATRIX(8,24,K^2*C^2,6,14)+  

EMATRIX(8,24,K^2,7,13)+EMATRIX(8,24,K^2,7,15);  

H7:EMATRIX(8,24,2*K^2*C^3,4,12)+EMATRIX(8,24,2*K^2*C^3,4,14)+  

EMATRIX(8,24,2*K^2*C,5,13)+EMATRIX(8,24,2*K^2*C,5,15)+  

EMATRIX(8,24,2*K^2*C^3,6,12)+EMATRIX(8,24,2*K^2*C^3,6,14)+  

EMATRIX(8,24,2*K^2*C,7,13)+EMATRIX(8,24,2*K^2*C,7,15);  

H2:EMATRIX(8,24,-3*C^4,1,9)+EMATRIX(8,24,-2*C^3,1,12)+  

EMATRIX(8,24,C^3,1,14)+EMATRIX(8,24,-3*C^2,2,10)+  

EMATRIX(8,24,2*C^3,2,11)+EMATRIX(8,24,C+C*SI^2,2,15)+  

EMATRIX(8,24,2*C^3,3,10)+EMATRIX(8,24,-2*C^2,3,15)+  

EMATRIX(8,24,-2*C^3,4,9)+EMATRIX(8,24,2*C^2,4,14)+  

EMATRIX(8,24,C^3,6,9)+EMATRIX(8,24,2*C^2,6,12)+  

EMATRIX(8,24,C^2,6,14)+EMATRIX(8,24,C*SI,6,15)+  

EMATRIX(8,24,C+C*SI^2,7,10)+EMATRIX(8,24,-2*C^2,7,11)+  

EMATRIX(8,24,C*SI,7,14)+EMATRIX(8,24,1+2*SI^2,7,15);

```

N1:ZEROMATRIX(8,8)\$

```

FOR II THRU 3 DO FOR JJ THRU 3 DO (PRINT(II,JJ),
(I1:(-12*II^2+44*II-16), J2:(12*JJ^2-52*JJ+64),
J1:(-12*JJ^2+44*JJ-16), I2:(12*II^2-52*II+64),
SUBI0:SUBMATRIX(H0,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
I2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
SUBJ0:SUBMATRIX(H0,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H0",II,JJ),
SUBI1:SUBMATRIX(H1,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
I2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
SUBJ1:SUBMATRIX(H1,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H1",II,JJ),
SUBI2:SUBMATRIX(H2,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
I2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
SUBJ2:SUBMATRIX(H2,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H2",II,JJ),
SUBI3:SUBMATRIX(H3,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
I2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
SUBJ3:SUBMATRIX(H3,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,

```

```

J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H3",II,JJ),

SUBI4:SUBMATRIX(H4,I1,J1-1,I1-2,I1-3,I1-4,I1-5,I1-6,I1-7,
I2,I2-1,I2-2,I2-3,I2-4,I2-5,I2-6,I2-7),
SUBJ4:SUBMATRIX(H4,I1,J1-1,I1-2,I1-3,I1-4,I1-5,I1-6,I1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H4",II,JJ),

SUBI5:SUBMATRIX(H5,I1,J1-1,I1-2,I1-3,I1-4,I1-5,I1-6,I1-7,
I2,I2-1,I2-2,I2-3,I2-4,I2-5,I2-6,I2-7),
SUBJ5:SUBMATRIX(H5,I1,J1-1,I1-2,I1-3,I1-4,I1-5,I1-6,I1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H5",II,JJ),

SUBI6:SUBMATRIX(H6,I1,J1-1,I1-2,I1-3,I1-4,I1-5,I1-6,I1-7,
I2,I2-1,I2-2,I2-3,I2-4,I2-5,I2-6,I2-7),
SUBJ6:SUBMATRIX(H6,I1,J1-1,I1-2,I1-3,I1-4,I1-5,I1-6,I1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H6",II,JJ),

SUBI7:SUBMATRIX(H7,I1,J1-1,I1-2,I1-3,I1-4,I1-5,I1-6,I1-7,
I2,I2-1,I2-2,I2-3,I2-4,I2-5,I2-6,I2-7),
SUBJ7:SUBMATRIX(H7,I1,J1-1,I1-2,I1-3,I1-4,I1-5,I1-6,I1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H7",II,JJ),

N1:N1+A[II,JJ]*
COL(L0,II).TQ.SUBJ0+(TQ.COL(L0,II)*SUBJ0+SUBI0.Q.ROW(L0T,JJ)),
PRINT("N1A",II,JJ),

N1:N1+DD[II,JJ]*
COL(L0,II).TQ.SUBJ2+(TQ.COL(L0,II)*SUBJ2+SUBI2.Q.ROW(L0T,JJ) +
COL(L1,II).TQ.SUBJ1+(TQ.COL(L1,II)*SUBJ1+SUBI1.Q.ROW(L1T,JJ) +
COL(L2,II).TQ.SUBJ0+(TQ.COL(L2,II)*SUBJ0+SUBI0.Q.ROW(L2T,JJ)),
PRINT("N1DD",II,JJ),

N1:N1+F[II,JJ]*
COL(L0,II).TQ.SUBJ4+(TQ.COL(L0,II)*SUBJ4+SUBI4.Q.ROW(L0T,JJ) +
COL(L1,II).TQ.SUBJ3+(TQ.COL(L1,II)*SUBJ3+SUBI3.Q.ROW(L1T,JJ) +
COL(L2,II).TQ.SUBJ2+(TQ.COL(L2,II)*SUBJ2+SUBI2.Q.ROW(L2T,JJ) +
COL(L3,II).TQ.SUBJ1+(TQ.COL(L3,II)*SUBJ1+SUBI1.Q.ROW(L3T,JJ) +
COL(L4,II).TQ.SUBJ0+(TQ.COL(L4,II)*SUBJ0+SUBI0.Q.ROW(L4T,JJ)),
PRINT("N1F",II,JJ),

N1:N1+H[II,JJ]*
COL(L0,II).TQ.SUBJ6+(TQ.COL(L0,II)*SUBJ6+SUBI6.Q.ROW(L0T,JJ) +
COL(L1,II).TQ.SUBJ5+(TQ.COL(L1,II)*SUBJ5+SUBI5.Q.ROW(L1T,JJ) +
COL(L2,II).TQ.SUBJ4+(TQ.COL(L2,II)*SUBJ4+SUBI4.Q.ROW(L2T,JJ) +
COL(L3,II).TQ.SUBJ3+(TQ.COL(L3,II)*SUBJ3+SUBI3.Q.ROW(L3T,JJ) +
COL(L4,II).TQ.SUBJ2+(TQ.COL(L4,II)*SUBJ2+SUBI2.Q.ROW(L4T,JJ) +

```

```
COL(L5,II).TQ.SUBJ1+(TQ.COL(L5,II))*SUBJ1+SUBI1.Q.ROW(L5T,II)+  
COL(L6,II).TQ.SUBJ0+(TQ.COL(L6,II))*SUBJ0+SUBI0.Q.ROW(L6T,II)).  
PRINT("N1H",II,II).
```

```
N1:N1+J[II,II]*(  
COL(L1,II).TQ.SUBJ7+(TQ.COL(L1,II))*SUBJ7+SUBI7.Q.ROW(L1T,II)+  
COL(L2,II).TQ.SUBJ6+(TQ.COL(L2,II))*SUBJ6+SUBI6.Q.ROW(L2T,II)+  
COL(L3,II).TQ.SUBJ5+(TQ.COL(L3,II))*SUBJ5+SUBI5.Q.ROW(L3T,II)+  
COL(L4,II).TQ.SUBJ4+(TQ.COL(L4,II))*SUBJ4+SUBI4.Q.ROW(L4T,II)+  
COL(L5,II).TQ.SUBJ3+(TQ.COL(L5,II))*SUBJ3+SUBI3.Q.ROW(L5T,II)+  
COL(L6,II).TQ.SUBJ2+(TQ.COL(L6,II))*SUBJ2+SUBI2.Q.ROW(L6T,II)+  
COL(L7,II).TQ.SUBJ1+(TQ.COL(L7,II))*SUBJ1+SUBI1.Q.ROW(L7T,II)).  
PRINT("N1J",II,II).
```

```
N1:N1+L[II,II]*(  
COL(L3,II).TQ.SUBJ7+(TQ.COL(L3,II))*SUBJ7+SUBI7.Q.ROW(L3T,II)+  
COL(L4,II).TQ.SUBJ6+(TQ.COL(L4,II))*SUBJ6+SUBI6.Q.ROW(L4T,II)+  
COL(L5,II).TQ.SUBJ5+(TQ.COL(L5,II))*SUBJ5+SUBI5.Q.ROW(L5T,II)+  
COL(L6,II).TQ.SUBJ4+(TQ.COL(L6,II))*SUBJ4+SUBI4.Q.ROW(L6T,II)+  
COL(L7,II).TQ.SUBJ3+(TQ.COL(L7,II))*SUBJ3+SUBI3.Q.ROW(L7T,II)).  
PRINT("N1L",II,II).
```

```
N1:N1+R[II,II]*(  
COL(L5,II).TQ.SUBJ7+(TQ.COL(L5,II))*SUBJ7+SUBI7.Q.ROW(L5T,II)+  
COL(L6,II).TQ.SUBJ6+(TQ.COL(L6,II))*SUBJ6+SUBI6.Q.ROW(L6T,II)+  
COL(L7,II).TQ.SUBJ5+(TQ.COL(L7,II))*SUBJ5+SUBI5.Q.ROW(L7T,II)).  
PRINT("N1R",II,II),
```

```
N1:N1+T[II,II]*(  
COL(L7,II).TQ.SUBJ7+(TQ.COL(L7,II))*SUBJ7+SUBI7.Q.ROW(L7T,II)).  
PRINT("N1T",II,II),
```

```
KILL(SUBJ0,SUBJ1,SUBJ2,SUBJ3,SUBJ4,SUBJ5,SUBJ6,SUBJ7),  
KILL(SUBI0,SUBI1,SUBI2,SUBI3,SUBI4,SUBI5,SUBI6,SUBI7));
```

```
SAVE("BEAMN1.SV",N1);
```

```
KILL(L0,L1,L2,L3,L4,L5,L6,L7,L0T,L1T,L2T,L3T,L4T,L5T,L6T,L7T)$  
KILL(H0,H1,H2,H3,H4,H5,H6,H7)$
```

```
N1SYM:ZEROMATRIX(8,8)$
```

```
FOR II THRU 8 DO FOR JJ:II THRU 8 DO N1SYM[II,II]:N1[II,II]$
```

```
PRINT("SYMMETRIC N1 FORMED")$
```

```
KILL(N1)$
```

```
N1:ZEROMATRIX(8,8)$
```

```
KILL(Q,TQ)$
```

```
*****
*****
/* THE FOLLOWING STATEMENTS GENERATE A FORTRAN STATEMENT FOR EACH
NONZERO */
/* ELEMENT OF SN1(I,J). THESE STATEMENTS ARE OF THE FORM      */
/*      SN1(2,2)=A(1,1).      */
/* EACH STATEMENT IS WRITTEN TO A SEPERATE FILE CALLED TT2XXX, WHERE XXX */
/* STARTS AT 001 FOR THE FIRST NONZERO ENTRY AND CONTINUES SEQUENTIALLY */
/* UNTIL ALL NONZERO ENTRIES THROUGH SN1(18,18) ARE GENERATED. THE      */
/* MACSYMA FUNTION GENTRAN WILL ALSO BREAK STATEMENTS EXCEEDING 800 INTO
*/
/* SHORTER EXPRESSIONS TO AVOID TOO MANY CONTINUATION LINES. MACSYMA      */
/* AUTOMATICALLY MAKES CONTINUATION LINES COMPLETE WITH A LEGAL      */
/* CHARACTER IN COLUMN 6.      */
*****
*****
```

FOR II THRU 8 DO FOR JJ:II THRU 8 DO
 N1[II,JJ]:FACTROUT(N1SYM[II,JJ],Q(1),Q(2),Q(3),Q(4),Q(5),Q(6),Q(7),Q(8))\$

FRAME(I,J):=CONCAT(TN,EV(8*(I-1)+J+1000))\$

FOR I THRU 8 DO FOR J:I THRU 8 DO
 (IF N1[I,J]#0 THEN (P1:1,GENTRAN(BMN1[EVAL(I),EVAL(J)]:EVAL(N1[I,J]),
 [EVAL(FRAME(I,J))]))\$

IF PT#1 THEN GENTRAN(PT:EVAL(PT),[TT2000])\$

CLOSEFILE();
 QUIT();

•

B.2 N_2 Inputs

WRITEFILE("BEAMN2.WF");

```
*****
*****
/* MACSYMA ROUTINE FOR ELEMENTAL CODE GENERATION BY S. A. SCHIMMELS      */
/* CREATED AS A PART OF AN AIR FORCE INSTITUTE OF TECHNOLOGY (AFIT)      */
/* MODIFIED BY CAPT DAN MILLER FOR 1-D BEAMS      */
/* PROGRAM IN AERONAUTICAL ENGINEERING --- MARCH 1993      */
/*          MACSYMA IS A REGISTERED TRADEMARK OF      */
/*          THE MASSACHUSETTS INSTITUTE OF TECHNOLOGY      */
*****
```

```

/*
/* FOR CURVED BEAM. CREATES ELEMENT
/* INDEPENDENT STIFFNESS ARRAYS N2 & N2S. */
***** */
***** */

/*
/* INITIALIZE MACSYMA PARAMETERS AND DECLARE VARIABLE PROPERTIES */
***** */

[DYNAMALLOC:TRUE,DISKGC:TRUE,DERIVABBREV:TRUE,POWERDISP:TRUE]$
DECLARE([K,C,SI],CONSTANT);

/*
/* GENERATE THE NONLINEAR ELEMENT-INDEPENDENT STIFFNESS ARRAY N2. */
***** */

/*
/* ASSEMBLE MATRIX N2 */
***** */

TQ: MATRIX([Q(1),Q(2),Q(3),Q(4),Q(5),Q(6),Q(7),Q(8)]);

Q: TRANSPOSE(TQ);

H0: EMATRIX(8,24,C^2,1,9)+EMATRIX(8,24,C,1,12)+  

    EMATRIX(8,24,1,2,10)+EMATRIX(8,24,-C,2,11)+  

    EMATRIX(8,24,-C,3,10)+EMATRIX(8,24,C^2,3,11)+  

    EMATRIX(8,24,C,4,9)+EMATRIX(8,24,1,4,12);  

H1: EMATRIX(8,24,C^2,1,12)+EMATRIX(8,24,C^2,1,14)+  

    EMATRIX(8,24,-C^2,2,11)+EMATRIX(8,24,1+SI^2,2,15)+  

    EMATRIX(8,24,-C^2,3,10)+EMATRIX(8,24,2*C^3,3,11)+  

    EMATRIX(8,24,-C-C*SI^2,3,15)+  

    EMATRIX(8,24,C^2,4,9)+EMATRIX(8,24,2*C,4,12)+  

    EMATRIX(8,24,C,4,14)+EMATRIX(8,24,C^2,6,9)+  

    EMATRIX(8,24,C,6,12)+  

    EMATRIX(8,24,SI,6,15)+EMATRIX(8,24,1+SI^2,7,10)+  

    EMATRIX(8,24,-C-C*SI^2,7,11)+EMATRIX(8,24,SI,7,14);  

H3: EMATRIX(8,24,2*C^5,1,9)+EMATRIX(8,24,K*C^2,1,12)+  

    EMATRIX(8,24,-2*C^4+K*C^2,1,14)+EMATRIX(8,24,2*C^3,2,10)+  

    EMATRIX(8,24,K,2,13)+EMATRIX(8,24,-2*C^2+K,2,15)+  

    EMATRIX(8,24,-K*C,3,13)+EMATRIX(8,24,-K*C,3,15)+  

    EMATRIX(8,24,K*C^2,4,9)+EMATRIX(8,24,2*K*C,4,12)+  

    EMATRIX(8,24,K*C,4,14)+EMATRIX(8,24,K,5,10)+  

    EMATRIX(8,24,-K*C,5,11)+EMATRIX(8,24,-2*C^4+K*C^2,6,9)+  

    EMATRIX(8,24,K*C,6,12)+EMATRIX(8,24,2*C^3,6,14)+  

    EMATRIX(8,24,-2*C^2+K,7,10)+EMATRIX(8,24,-K*C,7,11)+
```

```

EMATRIX(8,24,2*C,7,15);
H4:EMATRIX(8,24,K*C^3,1,12)+EMATRIX(8,24,K*C^3,1,14)+  

EMATRIX(8,24,K*C,2,13)+EMATRIX(8,24,K*C,2,15)+  

EMATRIX(8,24,-2*K*C^2,3,13)+EMATRIX(8,24,-2*K*C^2,3,15)+  

EMATRIX(8,24,K*C^3,4,9)+EMATRIX(8,24,4*K*C^2,4,12)+  

EMATRIX(8,24,3*K*C^2,4,14)+EMATRIX(8,24,K*C,5,10)+  

EMATRIX(8,24,-2*K*C^2,5,11)+EMATRIX(8,24,K+K*SI^2,5,15)+  

EMATRIX(8,24,K*C^3,6,9)+EMATRIX(8,24,3*K*C^2,6,12)+  

EMATRIX(8,24,2*K*C^2,6,14)+EMATRIX(8,24,K*C,7,10)+  

EMATRIX(8,24,-2*K*C^2,7,11)+EMATRIX(8,24,K+K*SI^2,7,13)+  

EMATRIX(8,24,2*K+2*K*SI^2,7,15);
H5:EMATRIX(8,24,-2*K*C^4,1,12)+EMATRIX(8,24,-2*K*C^4,1,14)+  

EMATRIX(8,24,-2*K*C^2,2,13)+EMATRIX(8,24,-2*K*C^2,2,15)+  

EMATRIX(8,24,-2*K*C^4,4,9)+EMATRIX(8,24,2*K*C^3,4,14)+  

EMATRIX(8,24,-2*K*C^2,5,10)+  

EMATRIX(8,24,2*K*C+2*K*C*SI^2,5,15)+  

EMATRIX(8,24,-2*K*C^4,6,9)+EMATRIX(8,24,2*K*C^3,6,12)+  

EMATRIX(8,24,4*K*C^3,6,14)+EMATRIX(8,24,-2*K*C^2,7,10)+  

EMATRIX(8,24,2*K*C+2*K*C*SI^2,7,13)+  

EMATRIX(8,24,4*K*C,7,15);
H6:EMATRIX(8,24,K^2*C^2,4,12)+EMATRIX(8,24,K^2*C^2,4,14)+  

EMATRIX(8,24,K^2,5,13)+EMATRIX(8,24,K^2,5,15)+  

EMATRIX(8,24,K^2*C^2,6,12)+EMATRIX(8,24,K^2*C^2,6,14)+  

EMATRIX(8,24,K^2,7,13)+EMATRIX(8,24,K^2,7,15);
H7:EMATRIX(8,24,2*K^2*C^3,4,12)+EMATRIX(8,24,2*K^2*C^3,4,14)+  

EMATRIX(8,24,2*K^2*C,5,13)+EMATRIX(8,24,2*K^2*C,5,15)+  

EMATRIX(8,24,2*K^2*C^3,6,12)+EMATRIX(8,24,2*K^2*C^3,6,14)+  

EMATRIX(8,24,2*K^2*C,7,13)+EMATRIX(8,24,2*K^2*C,7,15);
H2:EMATRIX(8,24,-3*C^4,1,9)+EMATRIX(8,24,-2*C^3,1,12)+  

EMATRIX(8,24,C^3,1,14)+EMATRIX(8,24,-3*C^2,2,10)+  

EMATRIX(8,24,2*C^3,2,11)+EMATRIX(8,24,C+C*SI^2,2,15)+  

EMATRIX(8,24,2*C^3,3,10)+EMATRIX(8,24,-2*C^2,3,15)+  

EMATRIX(8,24,-2*C^3,4,9)+EMATRIX(8,24,2*C^2,4,14)+  

EMATRIX(8,24,C^3,6,9)+EMATRIX(8,24,2*C^2,6,12)+  

EMATRIX(8,24,C^2,6,14)+EMATRIX(8,24,C*SI,6,15)+  

EMATRIX(8,24,C+C*SI^2,7,10)+EMATRIX(8,24,-2*C^2,7,11)+  

EMATRIX(8,24,C*SI,7,14)+EMATRIX(8,24,1+2*SI^2,7,15);

```

N2:ZEROMATRIX(8,8)\$

CA:1/2;
CB:1/3;
CC:2/3;

FOR II THRU 3 DO FOR JJ THRU 3 DO (PRINT(II,JJ),
(I1:(-12*II^2+44*II-16), J2:(12*JJ^2-52*JJ+64),
J1:(-12*JJ^2+44*JJ-16), I2:(12*II^2-52*II+64),

SUBI0:SUBMATRIX(H0,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
I2,I2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
SUBJ0:SUBMATRIX(H0,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,

```

J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H0",II,JJ),

SUBI1:SUBMATRIX(H1,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
I2,I2-1,I2-2,I2-3,I2-4,I2-5,I2-6,I2-7),
SUBJ1:SUBMATRIX(H1,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H1",II,JJ),

SUBI2:SUBMATRIX(H2,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
I2,I2-1,I2-2,I2-3,I2-4,I2-5,I2-6,I2-7),
SUBJ2:SUBMATRIX(H2,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H2",II,JJ),

SUBI3:SUBMATRIX(H3,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
I2,I2-1,I2-2,I2-3,I2-4,I2-5,I2-6,I2-7),
SUBJ3:SUBMATRIX(H3,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H3",II,JJ),

SUBI4:SUBMATRIX(H4,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
I2,I2-1,I2-2,I2-3,I2-4,I2-5,I2-6,I2-7),
SUBJ4:SUBMATRIX(H4,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H4",II,JJ),

SUBI5:SUBMATRIX(H5,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
I2,I2-1,I2-2,I2-3,I2-4,I2-5,I2-6,I2-7),
SUBJ5:SUBMATRIX(H5,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H5",II,JJ),

SUBI6:SUBMATRIX(H6,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
I2,I2-1,I2-2,I2-3,I2-4,I2-5,I2-6,I2-7),
SUBJ6:SUBMATRIX(H6,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H6",II,JJ),

SUBI7:SUBMATRIX(H7,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
I2,I2-1,I2-2,I2-3,I2-4,I2-5,I2-6,I2-7),
SUBJ7:SUBMATRIX(H7,J1,J1-1,J1-2,J1-3,J1-4,J1-5,J1-6,J1-7,
J2,J2-1,J2-2,J2-3,J2-4,J2-5,J2-6,J2-7),
PRINT("H7",II,JJ),

N2:N2+ A[II,JJ]*(SUBI0.Q.TQ.SUBJ0+CA*(TQ.SUBJ0.Q)*SUBI0),
PRINT("N2A",II,JJ),

N2:N2+DD[II,JJ]*(CB*(SUBI0.Q.TQ.SUBJ2+CA*(TQ.SUBI0.Q)*SUBJ2+
SUBI2.Q.TQ.SUBJ0+CA*(TQ.SUBI2.Q)*SUBJ0)+
SUBI1.Q.TQ.SUBJ1+CA*(TQ.SUBJ1.Q)*SUBI1),
PRINT("N2D",II,JJ),

```

```

N2:N2+ F[II,JJ]*(CB*(SUBI0.Q.TQ.SUBJ4+CA*(TQ.SUBI0.Q)*SUBJ4+
SUBI4.Q.TQ.SUBJ0+CA*(TQ.SUBI4.Q)*SUBJ0)+
CC*(SUBI1.Q.TQ.SUBJ3+CA*(TQ.SUBI1.Q)*SUBJ3+
SUBI3.Q.TQ.SUBJ1+CA*(TQ.SUBI3.Q)*SUBJ1)+
SUBI2.Q.TQ.SUBJ2+CA*(TQ.SUBJ2.Q)*SUBI2),
PRINT("N2F",II,JJ),

N2:N2+ H[II,JJ]*(CB*(SUBI0.Q.TQ.SUBJ6+CA*(TQ.SUBI0.Q)*SUBJ6+
SUBI6.Q.TQ.SUBJ0+CA*(TQ.SUBI6.Q)*SUBJ0)+
CC*(SUBI1.Q.TQ.SUBJ5+CA*(TQ.SUBI1.Q)*SUBJ5+
SUBI5.Q.TQ.SUBJ1+CA*(TQ.SUBI5.Q)*SUBJ1)+
CC*(SUBI2.Q.TQ.SUBJ4+CA*(TQ.SUBI2.Q)*SUBJ4+
SUBI4.Q.TQ.SUBJ2+CA*(TQ.SUBI4.Q)*SUBJ2)+
SUBI3.Q.TQ.SUBJ3+CA*(TQ.SUBJ3.Q)*SUBI3),
PRINT("N2H",II,JJ),

N2:N2+ J[II,JJ]*(CC*(SUBI1.Q.TQ.SUBJ7+CA*(TQ.SUBI1.Q)*SUBJ7+
SUBI7.Q.TQ.SUBJ1+CA*(TQ.SUBI7.Q)*SUBJ1)+
CC*(SUBI2.Q.TQ.SUBJ6+CA*(TQ.SUBI2.Q)*SUBJ6+
SUBI6.Q.TQ.SUBJ2+CA*(TQ.SUBI6.Q)*SUBJ2)+
CC*(SUBI3.Q.TQ.SUBJ5+CA*(TQ.SUBI3.Q)*SUBJ5+
SUBI5.Q.TQ.SUBJ3+CA*(TQ.SUBI5.Q)*SUBJ3)+
SUBI4.Q.TQ.SUBJ4+CA*(TQ.SUBJ4.Q)*SUBI4),
PRINT("N2J",II,JJ),

N2:N2+ L[II,JJ]*(CC*(SUBI3.Q.TQ.SUBJ7+CA*(TQ.SUBI3.Q)*SUBJ7+
SUBI7.Q.TQ.SUBJ3+CA*(TQ.SUBI7.Q)*SUBJ3)+
CC*(SUBI4.Q.TQ.SUBJ6+CA*(TQ.SUBI4.Q)*SUBJ6+
SUBI6.Q.TQ.SUBJ4+CA*(TQ.SUBI6.Q)*SUBJ4)+
SUBI5.Q.TQ.SUBJ5+CA*(TQ.SUBJ5.Q)*SUBI5),
PRINT("N2L",II,JJ),

N2:N2+ R[II,JJ]*(CC*(SUBI5.Q.TQ.SUBJ7+CA*(TQ.SUBI5.Q)*SUBJ7+
SUBI7.Q.TQ.SUBJ5+CA*(TQ.SUBI7.Q)*SUBJ5)+
SUBI6.Q.TQ.SUBJ6+CA*(TQ.SUBJ6.Q)*SUBI6),
PRINT("N2R",II,JJ),

N2:N2+ T[II,JJ]*(SUBI7.Q.TQ.SUBJ7+CA*(TQ.SUBJ7.Q)*SUBI7),
PRINT("N2T",II,JJ),

KILL(SUBJ0,SUBJ1,SUBJ2,SUBJ3,SUBJ4,SUBJ5,SUBJ6,SUBJ7),
KILL(SUBI0,SUBI1,SUBI2,SUBI3,SUBI4,SUBI5,SUBI6,SUBI7));
KILL(H0,H1,H2,H3,H4,H5,H6,H7)$

N2SYM:ZEROMATRIX(8,8)$

SAVE("BEAM-N2.SV",N2)$

FOR II THRU 8 DO FOR JJ:II THRU 8 DO N2SYM[II,JJ]:N2[II,JJ]$
```

```
PRINT("SYMMETRIC N2 FORMED")$
```

```
KILL(N2)$
```

```
N2:ZEROMATRIX(8,8)$
```

```
KILL(Q,TQ)$
```

```
******/  
******/  
/* THE FOLLOWING STATEMENTS GENERATE A FORTRAN STATEMENT FOR EACH  
NONZERO */  
/* ELEMENT OF SN2(I,J). THESE STATEMENTS ARE OF THE FORM */  
/* SN2(2,2)=A(1,1). */  
/* EACH STATEMENT IS WRITTEN TO A SEPERATE FILE CALLED TT2XXX, WHERE XXX */  
/* STARTS AT 001 FOR THE FIRST NONZERO ENTRY AND CONTINUES SEQUENTIALLY */  
/* UNTIL ALL NONZERO ENTRIES THROUGH SN2(18,18) ARE GENERATED. THE */  
/* MACSYMA FUNTION GENTRAN WILL ALSO BREAK STATEMENTS EXCEEDING 800 INTO */  
/*  
/* SHORTER EXPRESSIONS TO AVOID TOO MANY CONTINUATION LINES. MACSYMA */  
/* AUTOMATICALLY MAKES CONTINUATION LINES COMPLETE WITH A LEGAL */  
/* CHARACTER IN COLUMN 6. */  
******/  
******/
```

```
FRAME(I,J):=CONCAT(TN,EV(8*(I-1)+J+1000))$
```

```
FOR II:1 THRU 8 DO FOR JJ:II THRU 8 DO  
N2[II,JJ]:FACTROUT(N2SYM[II,JJ],Q(1),Q(2),Q(3),Q(4),Q(5),Q(6),Q(7),Q(8))$
```

```
FOR I:1 THRU 8 DO FOR J:I THRU 8 DO  
(IF N2[I,J]#0 THEN (PT:1,GENTRAN(BMN2[EVAL(I),EVAL(J)]:EVAL(N2[I,J]),  
[EVAL(FRAME(I,J))])))$
```

```
CLOSEFILE();  
QUIT();
```

```
.
```

Appendix C FORTRAN Program Description

C.1 Program Description

The attached code was initially developed by Creaghan. Modifications that were developed in chapter II were incorporated into the code. Most of the changes involved forming the element stiffness routines. The Riks and displacement control solution techniques were used as presented by Creaghan and required only minor modifications to fix bugs. The code subroutines are described and the code itself included for readers to examine the details of the theory implementation. Not all subroutines are listed, just the major routines that enable the reader to better understand the code.

1. *beam* - This is the main finite element program that calls the input routine (*rinput*), elasticity routine (*elast*) and which ever solution technique is specified.
2. *rinput* - Reads in program data from user defined file. Input information is echoed to the output file for easier confirmation of reading correct data.
3. *elast* - This routine calculates the elasticity terms for either isotropic or composite material.
4. *proces* - This is the main subroutine for doing displacement control solutions. It calls *stiff* to form element stiffness matrices, assembles them globally, then calls *solve* to find incremental solutions. *proces* increments displacement, applies boundary conditions, and calls *converge* to check for incremental convergence. When an increment is complete, *postpr* is called to do any required post processing.
5. *rikspr* - This is the main Riks method subroutine. It acts much the same as *proces* does for displacement control, but incrementing parameters are more complicated due to the

complicated nature of the technique. The logic used in Riks came directly from Tsai's code.

6. *stiff* - This subroutine calculates the element stiffness arrays. Routines are called to find \hat{K} , \hat{N}_1 and \hat{N}_2 . *stiff* then calls *shape* which evaluates the shape functions at the Gauss points. Five point Gauss quadrature takes place in *stiff* for each element stiffness array. The energy calculations from chapter III and appendix A were also conducted here since the shape functions were already evaluated and the integration was taking place. The biggest changes to the code to incorporate the current theory came from finding \hat{K} , \hat{N}_1 and \hat{N}_2 . The equations for each are shown in appendix A and involve thousands of manipulations. The MACSYMA files shown in appendix B were used to generate the calculations in these routines. If one examines the code, nearly 75% of the lines of code are in calculating these arrays.

7. *shape* - This subroutine evaluates the shape functions at each Gauss point and include the Jacobian terms.

8. *postpr* - This routine does any necessary post processing after convergence of a particular increment. Any resultant forces are generated here and placed in an output file with nodal displacements. It also generates a file called "plot" which contains (in column format) the displacement for the degree of freedom specified for force computation, the displacement for the degree of freedom two less than the dof specified for force compilation, and the force for the specified dof. For example, if dof 60 is vertical displacement at an arch crown (*w*) and is specified for force computation, then plot contains dof 60 in the first column, dof 58 in the second column (*v* of the crown in this case), and the crown vertical force in the third column. Nodal coordinates at each increment are placed in a file called "bshape". The coordinates are expressed in a Cartesian reference frame with the radius of curvature as the origin. For a flat beam, the end with the first node is the origin.

C.2 Data Input Format

The following provides the data format for each line of the input file. Each section represents a line in the file and each variable described for a particular line is separated by a comma. Variables are in italics.

1. *title* - problem title text string
2. *linear, isotro, isarch, ishape*:
 - a. *linear*: 0 for nonlinear problem, 1 for linear problem
 - b. *isotro*: 0 for composite, 1 for isotropic
 - c. *isarch*: 0 for straight beam, 1 for circular arch
 - d. *ishape*: 1 to compute node coordinates for straight beam or full arch, 2 if half the arch is being modeled
3. *inctyp, ninc, imax, kupdte, tol*:
 - a. *inctyp*: 1 for displacement control, 2 for Riks method
 - b. *ninc*: number of increments (load or displacement) desired
 - c. *imax*: maximum number of iterations per increment
 - d. *kupdte*: not used but fill with 1 or 0
 - e. *tol*: convergence tolerance for an increment. Usually .01 or less(vary for Riks)
4. *pincr, eiter, ttpi*: include this line only if *inctyp*=2 (Riks) and *linear*=0 (nonlinear)
 - a. *pincr*: initial load parameter (λ_0) try 0.1
 - b. *eiter*: N_i - estimate at the number of iterations per increment. Not usually an integer
 - c. *ttpi*: maximum load increment or decrement for an iteration (λ_{max})
5. *table(ninc)*: include only for *inctyp*=1 and *linear*=0 (nonlinear displacement control).
This is a list of *ninc* numbers that will be multiplied by nodal displacement to get the displacement for an increment.
6. *nelem*: number of elements in the mesh
7. *delem(nelem)*: list of element lengths, one length for each element

8. *nbndry*: number of nodes with specified displacement boundary conditions
9. *nbound(nbndry,5)*: one line for each node with specified displacement boundary conditions. The first number is the node number (each element has three nodes, but only end nodes are numbered, displacement cannot be specified at a mid node). The next four numbers describe whether or not the dofs are free or specified. 1=specified , 0 = free. Order of dofs is v, ψ, w and $w_{,2}$.
10. *vbond(ii)*: real prescribed displacements for those dofs fixed above in the order that they were specified. For displacement control, these values will be multiplied by incremental values in *table(ninc)*.
11. *ldtyp, distld, ldtyp*: this line was originally intended for distributed load, but is not used. Fill with 0, 0.0 as this input line is still required.
12. *nconc*: number of concentrated loads (or moments). Riks needs at least 1
13. *iconc(nconc)*: skip if *nconc*=0. dof numbers for concentrated loads. Middle node dof count here (9 dofs per element). The order is $v^l, \psi^l, w^l, w_{,2}^l, v^3, v^2, \psi^2, w^2, w_{,2}^2$
14. *vconc(nconc)*: values of loads at dofs above. skip if *nconc*=0
15. *ey, nu, ht, width*: include for isotropic material
 - a. *ey*: Young's modulus
 - b. *nu*: Poisson's ratio
 - c. *ht*: thickness
 - d. *width*: beam or arch width
16. *e1, e2, g12, nul2, g13, g23, width*: include for composite material
 - a. *e1*: Young's modulus along the fibers
 - b. *e2*: Young's modulus transverse to fibers
 - c. *g12*: shear modulus G_{12}
 - d. *nul2*: Poisson's ratio ν_{12}
 - e. *g13*: shear modulus G_{13}

f. g_{23} : shear modulus G_{23}

g. *width*: beam or arch width

17. *nplies, pthick*: include for composite

a. *nplies*: number of plies

b. *pthick*: ply thickness (same for all plies)

18. *theta(nplies)*: include for composite; list of ply orientation angles in degrees

19. *rad*: include for curved arch (*isarch*=1); arch radius of curvature

20. *nforc*: number of nodal resultant forces to calculate

21. *iforc(nforc)*: include only for *nforc*>0. dof numbers for force calculations

22. *nstres*: originally to find stress which is no longer included. Fill with a 0.

C.3 FORTRAN Code Listing

All areas changed for large rotation have a line drawn in the right column

```
program beam
c
c      See bottom of file for variable and subroutine listing.
c      This version is n1 n2 updated to include tangent function
c      remember -e compile option (will give 132 character lines)
c      implicit double precision (a-h,o-z)
c      character*64 gname
c
c
c      common/chac/gname,fname
c
c      common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
c      common/input/tol,table(250),delem(250),vbound(2500),distld,
c      . vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
c      . rad,linear,isotro,isarch,ishape,inctyp,ninc,imax,
c      . nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
c      . nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
c      . theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
c
c      common/stf/stif(9,9),elp(9),eln(9,9),eld(9)
c
c      common/proc/gstif(2500,9),gn(2500,9),gf(2500),gd(2500),vperm(2500)
c
c      . vpres(2500)
c
c      call rinput
c      call elast
c      if(inctyp.eq.1)call proces
c      if(inctyp.eq.2)call rikspr
c*****
c
c      VARIABLES FOR BSHELL
c
c      fname input file
c      gname output file
c      ae,de,      elasticity terms
c      fe,he,      elasticity terms
c      ej,el,      elasticity terms
c      re,te,      elasticity terms
c      as,ds,      elasticity terms
c      fs      elasticity term
c      ey      Young's modulus for isotropic case
c      enu      Poisson's ratio for isotropic case
c      ht      thickness of beam for isotropic case
c      e1,e2,      laminate material properties
c      g12,enu12, "
c      enu21,g13, "
c      g23      "
c      pthick     laminate ply thickness
c      nplies     number of plies in laminate
c      theta(20)  ply orientation angles
```

```

c      tol  convergence tolerance, percent
c      table(250)  displacement increment multiplicative
c                  factors
c      delem(250)  element lengths
c      vbound(2500)  values of prescribed displacement boundary
c                  conditions
c      distld  distributed load intensity
c      vconc(2500)  concentrated load values
c      rad  arch radius of curvature
c      linear  =1 for linear analysis, =0 for nonlinear
c      isotro  =1 for isotropic, =0 for laminate
c      isarch  =1 for arch, =0 for straight beam
c      ishape  =1 to print x,y coordinates for each node at each
c      increment
c                  when a full arch is represented output to file 'bshape'
c      inclod  =1 to increment load(NA), =0 increment displacement
c      ninc  total number of displacement increments
c      imax  maximum number of iterations per increment
c      nelem  total number of elements in model
c      nbndry  number of nodes with specified boundary conditions
c      nbound(250,5)  array of node numbers followed by 1's for
c                  fixed b.c.'s, zeros for unfixed
c      ldtyp =1 for distributed load, =0 no distributed load
c      nconc  total number of concentrated loads input
c      iconc(2500)  DOF's for specified loads
c      nforc  number of forces(including moments)to be solved for
c      iforc(2500)  DOF's at which to calculate forces
c      nstres  number of elements for stress calculation
c      istres(250)  element #'s for stress calculation
c      ibndry(2500)  DOF numbers for b.c.'s
c      idload(250)  elements with distributed load
c      coord(251)  coordinate of the nodes
c      width  beam or arch width
c      nnod  number of nodes
c*****
*****  

c
c      SUBROUTINES FOR BSHELL
c
c      rinput  reads in and echos input data
c      elast  computes elasticity terms
c      proces  drives the solution algorithm for displacement control
c      rikspr  drives the solution algorithm for Riks method
c      stiff  manages stiffness matrix computations
c      shape  computes shape function array dsf
c      beamk  computes constant stiffness array bmk
c      beamnl  computes linear stiffness array bmn1
c      beamn2  computes quadratic stiffness array bmn2
c      bndy  applies displacement boundary conditions
c      solve  solves simultaneous equations in banded array format
c      converge  checks solutions for convergence
c      postpr  computes nodal loads and sends to output file
c
c      end
c
c
c      subroutine rinput

```

```

c
      character*64 fname,gname
      character*4 title
      dimension title(20)
      implicit double precision (a-h,o-z)
c
      common/chac/gname,fname
      common/input/tol,table(250),delem(250),vbound(2500),distld,
. vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
. rad,linear,isotro,isearch,ishape,inctyp,ninc,imax,
. nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
. nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
. theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
c
      write(*,1000)
      read(*,1005) fname
      write(*,1010)
      read(*,1005) gname
      open(5,file=fname)
      open(6,file=gname,status='new')
      read(5,1015) title
      read(5,*) linear,isotro,isearch,ishape
      read(5,*) inctyp,ninc,imax,kupdte,tol
      if(linear.eq.0.and.inctyp.eq.2)read(5,*)pincr,eiter,ttpi
      if(linear.eq.0.and.inctyp.eq.1)read(5,*)(table(i),i=1,ninc)
      read(5,*)nelem
      read(5,*)(delem(i),i=1,nelem)
c
c      calculate nodal coordinates
c
      nnod=nelem+1
      coord(1)=0.0
      do 5 ii=2,nnod
      5 coord(ii)=coord(ii-1)+delem(ii-1)
      read(5,*)nbndry
      do 10 i=1,nbndry
      10 read(5,*)(nbound(i,j),j=1,5)
      ifdof=0
c
c      ifdof=counter for enumber of fixed dof's
c
      do 20 i=1,nbndry
      do 20 j=2,5
      if(nbound(i,j).eq.0)goto 20
      ifdof=ifdof+1
      ibndry(ifdof)=(nbound(i,1)-1)*5 + (j-1)
      20 continue
      read(5,*)(vbound(i),i=1,ifdof)
      read(5,*)ldtyp,distld
      if(ldtyp.eq.1)read(5,*)ndload
      if(ldtyp.eq.1)read(5,*)(idload(i),i=1,ndload)
      read(5,*)nconc
      if(nconc.ne.0)read(5,*)(iconc(i),i=1,nconc)
      if(nconc.ne.0)read(5,*)(vconc(i),i=1,nconc)
      if(isotro.eq.1)read(5,*)ey,enu,ht,width
      if(isotro.eq.0)read(5,*)e1,e2,g12,enu12,g13,g23,width
      if(isotro.eq.0)read(5,*)nplies,pthick
      if(isotro.eq.0)read(5,*)(theta(i),i=1,nplies)

```

```

if(isarch.eq.1) read(5,*) rad
read(5,*) nforc
if(nforc.ne.0) read(5,*) (iforc(i),i=1,nforc)
read(5,*) nstres
if(nstres.ne.0) read(5,*) (istres(i),i=1,nstres)

C
C      Echo the input to the output file
C

      write(6,1015) title
      if(isarch.eq.1) write(6,1020)
      if(isarch.eq.0) write(6,1025)
      if(linear.eq.1) write(6,1030)
      if(linear.eq.0) write(6,1035)
      if(isotro.eq.1) write(6,1040)
      if(isotro.eq.0) write(6,1045)
      if(ishape.eq.1) write(6,1050)
      if(inctyp.eq.1) write(6,1060)
      if(inctyp.eq.2) write(6,1055)
      write(6,1065) ninc
      write(6,1070) imax
      write(6,1075) tol
      if(inctyp.eq.2) write(6,1076) pincr,eiter,ttpi
      if(inctyp.eq.1) write(6,1078)
      if(inctyp.eq.1) write(6,1080) (table(i),i=1,ninc)
      write(6,1085) nelem
      write(6,1090)
      write(6,1095) (coord(i),i=1,nnod)
      write(6,1100)
      write(6,1105)
      do 30 i=1,nbndry
30   write(6,1110) (nbound(i,j),j=1,5)
      write(6,1115) ifdof
      write(6,1120) (ibndry(i),i=1,ifdof)
      write(6,1095) (vbound(i),i=1,ifdof)
      if(ldtyp.eq.1) write(6,1125) distld
      if(ldtyp.eq.1) write(6,1130) (idload(i),i=1,ndload)
      if(nconc.ne.0) write(6,1135)
      if(nconc.ne.0) write(6,1120) (iconc(i),i=1,nconc)
      if(nconc.ne.0) write(6,1095) (vconc(i),i=1,nconc)
      if(isotro.eq.1) write(6,1140) ey,enu,ht,width
      if(isotro.eq.0) write(6,1145) e1,e2,g12,enu12,g13,g23,width
      if(isotro.eq.0) write(6,1150) nplies,pthick
      if(isotro.eq.0) write(6,1155) (theta(i),i=1,nplies)
      if(isarch.eq.1) write(6,1160) rad
      write(6,1165) (iforc(i),i=1,nforc)
      write(6,1170) (istres(i),i=1,nstres)
      close(5)
      close(6)

C
C      F   O   R   M   A   T   S
C

1000 format('Enter your input file name.')
1005 format(A)
1010 format('Enter your output file name.')
1015 format(20a4)
1020 format(/,1x,'Element type: arch')
1025 format(/,1x,'Element type: straight beam')

```

```

1030  format(/,1x,'Analysis type: linear')
1035 format(/,1x,'Analysis type: nonlinear')
1040 format(/,1x,'Material type: isotropic')
1045 format(/,1x,'Material type: laminate')
1050 format(/,1x,'Printout of nodal x,y coordinates requested')
1055 format(/,1x,'Riks method specified')
1060 format(/,1x,'Displacement control method specified')
1065 format(/,1x,'Increments specified:',2x,i3)
1070 format(/,1x,'Maximum iterations specified:',2x,i3)
1075 format(/,1x,'Percent convergence tolerance:',2x,d12.5)
1076 format(/,1x,'pincr=',2x,d12.5,2x,'eiter=',2x,d12.5,2x,
.      'tppi=',2x,d12.5)
1078 format(/,1x,'Displacement Increment Table')
1080 format(8(2x,d12.5))
1085 format(/,1x,'Number of elements:',2x,i3)
1090 format(/,1x,'Nodal Coordinates:')
1095 format(8(2x,d12.5))
1100 format(/,1x,'DISPLACEMENT BOUNDARY CONDITIONS, 1=PRESCRIBED,
X0=FREE')
1105 format(/,4X,'NODE      V  PSI-S    W  W-S  ')
1110 format(4x,i4,1x,4(i3,2x))
1115 format(/,1x,'NUMBER OF PRESCRIBED DISPLACEMENTS:',
.      i5,/,1x,'SPECIFIED DISPLACEMENT DOF AND THIER
.      VALUES FOLLOW:')
1120 format(16i5)

1125 format(/,1x,'Distributed Load Intensity:',2x,d12.5)
1130 format(/,1x,'Elements with distributed load:',/,1x,16i5)
1135 format(/,1x,'DOF and specified concentrated loads follow:')
1140 format(/,1x,'Isotropic material properties ey, enu, ht, width:'
.      ,/,1x,4d12.5)
1145 format(/,1x,'Composite material properties e1, e2, g12, enu12,
.      g13, g23, width:',/,1x,7d12.5)
1150 format(/,1x,'Number of plies:',2x,i3,2x,'Ply thickness:',2x,
.      d12.5)
1155 format(/,1x,'Ply orientation angles:',/,1x,8(2x,d12.5))
1160 format(/,1x,'Radius of curvature:',2x,d12.5)
1165 format(/,1x,'DOFs for equivalent load calculation:',/,
.      1x,16i5)
1170 format(/,1x,'Elements for stress calculation:',/,1x,16i5)
1175 format(/,1x,i5)
      return
      end

c
c
c
      subroutine elast
c
      implicit double precision (a-h,o-z)
c
      dimension qbar(3,3),rtheta(20)
c
c
      character*64 gname
      common/chac/gname, fname
      common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
      common/input/tol,table(250),delem(250),vbound(2500),distld,
.      vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,

```

```

. rad,linear,isotro,isarch,ishape,inctyp,ninc,imax,
. nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
. nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
. theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi

c
c
c Isotropic case
c
c write(6,1000)e1,e2,g12,enu12,enu21,g13,g23,pthick
if(isotro.eq.0)goto 100
gs=ey/(2*(1+enu))
denom=1.-enu**2
q11=ey/denom
q12=enu*ey/denom
q22=q11
q2hat=q22-(q12**2/q11)
qs4=gs
ae=q2hat*ht
de=q2hat*ht**3/(3*2.**2)
fe=q2hat*ht**5/(5*2.**4)
he=q2hat*ht**7/(7*2.**6)
ej=q2hat*ht**9/(9*2.**8)
el=q2hat*ht**11/(11*2.**10)
re=q2hat*ht**13/(13*2.**12)
te=q2hat*ht**15/(15*2.**14)
as=qs4*ht
ds=qs4*ht**3/(3*2.**2)
fs=qs4*ht**5/(5*2.**4)
goto 200
c
c Laminate case
c
100 ht=pthick*nplies
enu21=e2*enu12/e1
denom=1.-enu12*enu21
q11=e1/denom
q12=enu12*e2/denom
q22=e2/denom
c*****
c calculate the elasticity matrices *
c
c remem that the z axis points down, *
c however, the first ply is the top ply, ie, *
c the ply with the most negative z !!! *
c*****
c
c initialize elasticity terms
c
ae=0.
de=0.
fe=0.
he=0.
ej=0.
el=0.
re=0.
te=0.
as=0.

```

```

ds=0.
fs=0.
      do 45 ii=1,nplies
45      rtheta(ii)=theta(ii)*3.14159265/180.
      do 50 kk=1,nplies

      qbar(1,1)=q11*(cos(rtheta(kk))**4)+2*q12*(sin(rtheta(kk))**2)*
.    (cos(rtheta(kk))**2)+q22*(sin(rtheta(kk))**4)

      qbar(1,2)=(q11+q22)*(sin(rtheta(kk))**2)*(cos(rtheta(kk))**2)+
.    q12*(sin(rtheta(kk))**4+cos(rtheta(kk))**4)

      qbar(2,2)=q11*(sin(rtheta(kk))**4)+2*q12*(sin(rtheta(kk))**2)*
.    (cos(rtheta(kk))**2)+q22*cos(rtheta(kk))**4
      qs4=g13*dcos(rtheta(kk))**2+g23*dsin(rtheta(kk))**2
      q2hat=qbar(2,2)-(qbar(1,2)**2/qbar(1,1))
      zl=(kk*1. - nplies*.5)*pthick
      zu=zl-pthick
      ae=ae + q2hat*pthick
      de=de + q2hat*(zl**3-zu**3)/3.
      fe=fe + q2hat*(zl**5-zu**5)/5.
      he=he + q2hat*(zl**7-zu**7)/7.
      ej=ej + q2hat*(zl**9-zu**9)/9.
      el=el + q2hat*(zl**11-zu**11)/11.
      re=re + q2hat*(zl**13-zu**13)/13.
      te=te + q2hat*(zl**15-zu**15)/15.
      as=as+qs4*pthick
      ds=ds+qs4*(zl**3-zu**3)/3.
      fs=fs+qs4*(zl**5-zu**5)/5.
50  continue
c 200 open(6,fiel=gname,status='old')
200 write(6,1000)ae,de,fe,he,ej,el,re,te,as,ds,fs
c  close(6)
1000 format(/,1x,'Elasticity terms:',//,1x,8(2x,d12.5))
      return
      end

c
c
c      subroutine proces
c      implicit double precision (a-h,o-z)
c      character*64 gname
c
c
c      common/chac/gname,fname
c
c      common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
c
c      common/input/tol,table(250),delem(250),vbound(2500),distld,
.    vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
.    rad,linear,isotro,isarch,ishape,inctyp,ninc,imax,
.    nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
.    nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
.    theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
      common/stf/stif(9,9),elp(9),eln(9,9),eld(9)
c

```

```

common/proc/gstif(2500,9),gn(2500,9),gf(2500),gd(2500),vperm(2500)

      vpres(2500)

c
      ndof=nnod*4+nelem
      ncount=1
      icount=1
      do 1 ii=1,ndof
1      gd(ii)=0.0d0
      do 2 ii=1,nbndry*5
2      vpres(ii)=0.0d0
      vperm(ii)=vbound(ii)

c
c      start new increment or iteration/
c      zero out global stiffness matrices and global force
c      vector
c
3      do 5 ii=1,ndof
      gf(ii)=0.0d0
      do 5 jj=1,9
      gstif(ii,jj)=0.0d0
5      gn(ii,jj)=0.0d0
      kcall=0

c
c      increment prescribed displacement for displacement control
c
      if(linear.eq.1)goto 9
      if(icount.ne.1)goto 9
      do 7 ii=1,nbndry*5
      if(ncount.eq.1)vbound(ii)=vperm(ii)*table(1)
7      if(ncount.gt.1)vbound(ii)=vperm(ii)*(table(ncount)-
           table(ncount-1))

c
c      loop over all elements for stiffness and forces
c
9      do 30 ielem=1,nelem
      do 10 ii=1,9
10      eld(ii)=gd(ii+(ielem-1)*5)

c
      kcall=kcall+1
      call stiff(ielem,icount,ncount,kcall)

c
c      Assemble global stiffness array, gstif, global equilibrium
c      stiffness, gn, in banded form. Half-bandwidth=9. Also
c      assemble global force vector, gf.

c
      nr=(ielem-1)*5 + 1
      do 30 jj=0,8
      gf(nr+jj)=gf(nr+jj)+elp(jj+1)
      do 30 kk=1,9-jj
      gstif(nr+jj,kk)=gstif(nr+jj,kk)+stif(jj+1,kk+jj)
      if(linear.eq.1)goto 30
      if(icount.eq.1 .and. ncount.eq.1)goto 30
      gn(nr+jj,kk)=gn(nr+jj,kk)+eln(jj+1,kk+jj)
30      continue

c
c      impose force boundary conditions
c      at this point, gf=R

```

```

c
      if(nconc.eq.0)goto 45
      do 40 ii=1,nconc
          nb=iconc(ii)
      40  gf(nb)=gf(nb)+vconc(ii)
      45  continue

c
c      calculate the residual force vector for nonlinear
c      analysis. -[gn]*(gd)+R=-[k+n1/2+n2/3]*{q}+R=gf
c
      if(icont.eq.1)goto 65
      do 60 ii=1,ndof
          add=0.
          do 50 kk=1,ii-1
              if(ii-kk+1 .gt. 9)goto 50
              add=add+gn(kk,ii-kk+1)*gd(kk)
      50  continue
      res=0.
      do 55 jj=1,9
          if(jj+ii-1 .gt. ndof)goto 55
          res=res + gn(ii,jj)*gd(jj+ii-1)
      55  continue

c
c      add to existing gf which already contains R
c
      gf(ii)=gf(ii)-res-add
      60 continue
      65 continue

c
c      impose displacement boundary conditions
c
      if(icont.eq.1)call bndy(ndof,gstif,gf,nbndry,ibndry,vbound)
      if(icont.gt.1)call bndy(ndof,gstif,gf,nbndry,ibndry,vpres)

c
c      solve system of equations, result in gf
c
      call solve(ndof,gstif,gf,0,detm,detm1)

c
c      update total displacement vector gd
c
      do 70 ii=1,ndof
      70  gd(ii)=gd(ii)+gf(ii)
          if(linear.eq.1)goto 80
          call converge(ndof,ncon,icont,tol,imax)

c
c      if no convergence (ncon=0) start next iteration
c
      if(ncon.eq.0)goto 3
      80  continue
      if(ncon.eq.1 .and. ncount.le.ninc)then
          call postpr(icont,ncount,kcall,ndof)
          if(ncount.eq.ninc)stop
          ncount=ncount+1
          icount=1
          goto 3
      endif
      return
  end

```

```

c
c subroutine bndy(ndof,s,sl,ndum,idum,vdum)
c
c.....subroutine used to impose boundary conditions on banded equations
c......
c
c implicit double precision (a-h,o-z)
dimension s(2500,9),sl(2500)
dimension idum(ndum*5),vdum(ndum*5)
do 300 nb = 1, ndum*5
ie = idum(nb)
sval = vdum(nb)
it=8
i=ie-9
do 100 ii=1,it
i=i+1
if (i .lt. 1) go to 100
j=ie-i+1
sl(i)=sl(i)-s(i,j)*sval
s(i,j)=0.0
100 continue
s(ie,1)=1.0
sl(ie)=sval
i=ie
do 200 ii=2,9
i=i+1
if (i .gt. ndof) go to 200
sl(i)=sl(i)-s(ie,ii)*sval
s(ie,ii)=0.0
200 continue
300 continue
return
end
c
c
c
c subroutine converge(ndof,ncon,icount,tol,imax)
c......
c checks for convergnece using global displacement criterion
c......
c implicit double precision (a-h,o-z)
common/proc/gstif(2500,9),gn(2500,9),gf(2500),gd(2500),vperm(2500)
'
.      vpres(2500)
c
rcurr=0.
do 10 m=1,ndof
10 rcurr=rcurr + gd(m)*gd(m)
if(icount.eq.1)rinit=rcurr
if(icount.eq.1)ncon=0
if(icount.eq.1)goto 20
c new criteria
ratio=100. * abs(sqrt(rcurr)-sqrt(pvalue))/sqrt(rinit)
if(ratio.le.tol)ncon=1
20 pvalue=rcurr
write(*,100)ncon,ratio,rinit,rcurr

```

```

100 format(1x,'ncon= ',i3,3x,'ratio= ',d14.6,' rinit= ',d14.6,
      x           ' rcurr= ',d14.6)
      if(icount.eq.imax)write(6,200)
      if(icount.eq.imax)stop
200 format(1x,'icount equals imax')
      if(ncon.eq.0)icount=icount+1
      return
      end

c
c
c
c      subroutine rikspr
c
c      implicit double precision (a-h,o-z)
c
c      character*64 gname
c
c
c      common/chac/gname,fname
c
c      common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
c
c      common/input/tol,table(250),delem(250),vbound(2500),distld,
c      . vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
c      . rad,linear,isotro,isarch,ishape,inctyp,ninc,imax,
c      . nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
c      . nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
c      . theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
c      common/stf/stif(9,9),elp(9),eln(9,9),eld(9)
c
c      common/proc/gstif(2500,9),gn(2500,9),gf(2500),gd(2500),vperm(2500)
c
c      . vpres(2500)
c
c      dimension
gld(2500),gld0(2500),gld1(2500),gdis(2500),gsti00(2500,9),
      . gf0(2500),gd00(2500)
      ndof=nnod*4+nelem
      ncount=1
      icount=1
      iicut=0
      do 1 ii=1,ndof
1      gd(ii)=0.0d0
      do 2 ii=1,nbndry*5
      vpres(ii)=0.0d0
2      vperm(ii)=vbound(ii)
c
c      start new increment or iteration/
c      zero out global stiffness matrices and global force
c      vector
c
      tpincr=0.0
      if(ncount.eq.1)goto 2993
c
c      start new increment
c
3      if(iicut.eq.0)dss=dss*eiter/icount
2993 icount=1

```

```

        do 2992 ii=1,ndof
        gld0(ii)=0.0d0
2992 gd00(ii)=gd(ii)
c
c      start new iteration
c
4      do 5 ii=1,ndof
        gf0(ii)=0.0d0
        do 5 jj=1,9
        gstif(ii,jj)=0.0d0
5      gn(ii,jj)=0.0d0
        kcall=0
c
c      increment prescribed displacement for displacement control
c
c      if(linear.eq.1)goto 9
c      if(icount.ne.1)goto 9
c      do 7 ii=1,nbndry*5
c      if(ncount.eq.1.and.iicut.eq.0)vbound(ii)=vperm(ii)*table(1)
c7      if(ncount.gt.1.or.iicut.gt.0)vbound(ii)=vperm(ii)*(table(ncount)-
c      .      table(ncount-1))
c
c      loop over all elements for stiffness and forces
c
9      do 30 ielem=1,nelem
        do 10 ii=1,9
10      eld(ii)=gd(ii+(ielem-1)*5)
c
        kcall=kcall+1
        call stiff(ielem,icount,ncount,kcall)
c
c      Assemble global stiffness array, gstif, global equilibrium
c      stiffness, gn, in banded form. Half-bandwidth=9. Also
c      assemble global force vector, gf.
c
        nr=(ielem-1)*5 + 1
        do 30 jj=0,8
        gf0(nr+jj)=gf0(nr+jj)+elp(jj+1)
        do 30 kk=1,9-jj
        gstif(nr+jj,kk)=gstif(nr+jj,kk)+stif(jj+1,kk+jj)
        if(linear.eq.1)goto 30
        if(icount.eq.1 .and. ncount.eq.1 .and.iicut.eq.0)goto 30
        gn(nr+jj,kk)=gn(nr+jj,kk)+eln(jj+1,kk+jj)
30      continue
c
c      impose force boundary conditions
c      at this point, gf=R
c
        if(nconc.eq.0)goto 45
        do 40 ii=1,nconc
        nb=iconc(ii)
40      gf0(nb)=gf0(nb)+vconc(ii)
45      continue
        do 47 ii=1,ndof
47      gf(ii)=gf0(ii)
        do 48 ii=1,ndof
        do 48 jj=1,9
48      gsti00(ii,jj)=gstif(ii,jj)

```

```

call bndy(ndof,gstif,gf,nbndry,ibndry,vbound)
c
call solve(ndof,gstif,gf,0,detm,detml)
dss0=0.0
do 49 ii=1,ndof
gdis(ii)=gf(ii)
49 dss0=dss0+gf(ii)*gf(ii)
if(icount.ne.1) go to 144
    detml=detm2
    detm2=detm
    if(ncount.eq.1.and.detm.lt.0.0 .and.iicut.eq.0) pincr=-pincr
    if(ncount.eq.1.and.iicut.eq.0) dss=pincr*dsqrt(dss0)
    if(ncount.ne.1.or.iicut.gt.0) pincr=
dss/dsqr(dss0)*detm*detml
    .          *pincr1/dabs(pincr1)
c
c      attempt at offloading at bifurcation points
c
c      if(iicut.eq.1)pincr=-pincr
c
pincr1=pincr

    prs=0.0
    do 142 ii=1,ndof
142    prs=prs+gf0(ii)*gld(ii)
    stifpa=pincr*prs
    do 143 ii=1,ndof
143    gld(ii)=pincr*gdis(ii)
144 continue
c
c      calculate the residual force vector for nonlinear
c      analysis. -[gn]*{gd}+R=-[k+n1/2+n2/3]*{q}+R=gf
c

if(icount.eq.1)goto 69
    do 60 ii=1,ndof
    add=0.
    do 50 kk=1,ii-1
    if(ii-kk+1 .gt. 9)goto 50
    add=add+gn(kk,ii-kk+1)*gd(kk)
50    continue
    res=0.
    do 55 jj=1,9
    if(jj+ii-1 .gt. ndof)goto 55
    res=res + gn(ii,jj)*gd(jj+ii-1)
55    continue
c
c      add to existing gf which already contains R
c
    gf(ii)=gf0(ii)*(pincr+tpincr)-res-add
60 continue
65 continue
c
c      impose displacement boundary conditions
c
call bndy(ndof,gsti00,gf,nbndry,ibndry,vbound)
c      if(icount.gt.1)call bndy(ndof,gstif,gf,nbndry,ibndry,vpres)
c

```

```

c      solve system of equations, result in gf
c      call solve(ndof,gstif, gf, 1, detm, detml)
c      through line 69 copied from Tsai's program
c
c      a1=dss0
c      a2=0.0
c      a3=0.0
c      do 147 ii=1,ndof
c         a2=a2+(gld(ii)+gf(ii))*gdis(ii)
c 147   a3=a3+gf(ii)*(2.0*gld(ii)+gf(ii))
c         d12=a2*a2-a1*a3
c         write(6,*) d12,a1,a2,a3
c
c         if(d12.lt.0.0)then
c
c            deal with complex roots  by cutting the search
c            radius (dss) in half
c
c            do 2991 ii=1,ndof
c 2991   gd(ii)=gd00(ii)
c            iicut=iicut + 1
c            if(iicut.gt.10)then
c              write(6,3000)
c              stop
c            endif
c            dss=dss/2.0
c            goto 3
c            endif
c            iicut=0
c            dpinc1=(-a2+dsqrt(d12))/a1
c            dpinc2=(-a2-dsqrt(d12))/a1
c            thetal=0.0
c            theta2=0.0
c            do 148 ii=1,ndof
c              gld0(ii)=gld(ii)
c              gld(ii)=gld(ii)+gf(ii)+dpinc1*gdis(ii)
c              gld1(ii)=gld0(ii)+gf(ii)+dpinc2*gdis(ii)
c              thetal=thetal+gld0(ii)*gld(ii)
c              theta2=theta2+gld0(ii)*gld1(ii)
c 148   continue
c            write(6,*) thetal,theta2
c            thet12=thetal*theta2
c            if(thet12.gt.0.0) go to 149
c            dpincr=dpinc1
c            if(theta2.gt.0.0) call chsign(gld,gld1,dpincr,dpinc2,ndof)
c            go to 150
c 149   dpib=-a3/(a2*2.0)
c            dpin1=dabs(dpib-dpinc1)
c            dpin2=dabs(dpib-dpinc2)
c            dpincr=dpinc1
c            if(dpin2.lt.dpin1) call chsign(gld,gld1,dpincr,dpinc2,ndof)
c 150   pincr=pincr+dpincr
c 69   continue
c
c      update total displacement vector gd

```

```

c
do 70 ii=1,ndof
70 gd(ii)=gd(ii)+gld(ii)-gld0(ii)
if(linear.eq.1)goto 80
call converge(ndof,ncon,icount,tol,imax)
c
c      if no convergence (ncon=0) start next iteration
c
if(ncon.eq.0)goto 4
80      continue
if(ncon.eq.1 .and. ncount.le.ninc)then
      call postpr(icount,ncount,kcall,ndof)
      if(ncount.eq.ninc)stop
      ncount=ncount+1
      tpincr=tpincr+pincr
      goto 3
      endif
3000  format(1x,'More than 10 consecutive imaginary roots')
3010 format(1x,i2)
      return
      end
c
c
c      subroutine solve(ndof,band,rhs,ires,detm,detml)
c
c      solve a banded symmetric system of equations
c
c
implicit double precision (a-h,o-z)
dimension band(2500,9),rhs(2500)
meqns=ndof-1
if(ires.gt.0)goto 90
do 500 npiv=1,meqns
c      print*,'npiv= ',npiv
      npivot=npiv+1
      lstsub=npiv+9-1
      if(lstsub.gt.ndof) lstsub=ndof
      do 400 nrow=npivot,lstsub
c      invert rows and columns for row factor
      ncol=nrow-npiv+1
      factor=band(npiv,ncol)/band(npiv,1)
      do 200 ncol=nrow,lstsub
      icol=ncol-nrow+1
      jcol=ncol-npiv+1
200  band(nrow,icol)=band(nrow,icol)-factor*band(npiv,jcol)
400  rhs(nrow)=rhs(nrow)-factor*rhs(npiv)
500  continue
      detm=1.0
      detml=0.0
      do 600 ii=1,ndof
c      write (*,*) 'BAND(',ii,'1)=',band(ii,1)
      detml=detml+dlog10(dabs(band(ii,1)))
600  detm=detm*band(ii,1)/dabs(band(ii,1))
      go to 101
90  do 100 npiv=1,meqns
      npivot=npiv+1
      lstsub=npiv+9-1

```

```

        if(lstsub.gt.ndof) lstsub=ndof
        do 110 nrow=npivot,lstsub
        ncol=nrow-npiv+1
        factor=band(npiv,ncol)/band(npiv,1)
110  rhs(nrow)=rhs(nrow)-factor*rhs(npiv)
100  continue
c      back substitution
101  do 800 ijk=2,ndof
        npiv=ndof-ijk+2
        rhs(npiv)=rhs(npiv)/band(npiv,1)
        lstsub=npiv-9+1
        if(lstsub.lt.1) lstsub=1
        npivot=npiv-1
        do 700 jki=lstsub,npivot
        nrow=npivot-jki+lstsub
        ncol=npiv-nrow+1
        factor=band(nrow,ncol)
700  rhs(nrow)=rhs(nrow)-factor*rhs(npiv)
800  continue
        rhs(1)=rhs(1)/band(1,1)
        return
        end
c
c
c
        subroutine chsign(gld,gld1,dpincr,dpinc2,ndof)
        implicit double precision (a-h,o-z)
        dimension gld(2500),gld1(2500)
        do 100 i=1,ndof
100  gld(i)=gld1(i)
        dpincr=dpinc2
        return
        end
c
c
c
        subroutine stiff(ielem,icount,ncount,kcall)
c
        implicit double precision (a-h,o-z)
        character*64 gname
        common/chac/gname,fname
c
        common/input/tol,table(250),delem(250),vbound(2500),distld,
.  vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
.  rad,linear,isotro,isearch,ishape,inctyp,ninc,imax,
.  nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
.  nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
.  theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
c
        common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
c
        common/stf/stif(9,9),elp(9),eln(9,9),eld(9)
c
        common/shp/dsf(7,9)
c
        dimension bmk(7,7),bmn1(7,7),bmn2(7,7),
.  gauss4(4),wt4(4),gauss7(7),wt7(7),q(7),dsftr(9,7),
.  pkt(7,7),pkn(7,7),pktd(7,9),pknd(7,9),gauss5(5),wt5(5)

```

```

c
  data gauss4/0.8611363115d0,0.3399810435d0,-0.3399810435d0,
.  -0.8611363115d0/
  data wt4/0.3478548451d0,0.6521451548d0,0.6521451548d0,
.  0.3478548451d0/
  data gauss5/0.9061798459d0,0.5384693101d0,0.0d0,-0.5384693101d0,
.  -0.9061798459d0/
  data wt5/0.2369268851d0,0.4786286705d0,0.5688888889d0,
.  0.4786286705d0,0.2369268851d0/
  data gauss7/0.9491079123d0,0.7415311856d0,0.4058451513d0,
.  0.0d0,-0.4058451513d0,-0.7415311856d0,-0.9491079123d0/
  data wt7/0.1294849662d0,0.2797053915d0,0.3818300505d0,
.  0.4179591836d0,0.3818300505d0,0.2797053915d0,0.1294849662d0/
c
c  initialize stiffness arrays and load array
c
  do 10 ii=1,9
    elp(ii)=0.0
    do 10 jj=1,9
      stif(ii,jj)=0.0
  10 eln(ii,jj)=0.0
c
c  set number of gauss points for interpolation
c
  ngp=5
  if(ncount.eq.1 .and. ictcount.eq.1)ngp=4
  if(linear.eq.1)ngp=4
c
  ek1=-4./(3.*ht**2)
  if (isearch.eq.1) p1=1./rad
  if(ncount.eq.1 .and. ictcount.eq.1 .and. kcall.eq.1 .and.
isearch.eq.1)
    .  call beamk(bmk,ek1,p1)
    if(ncount.eq.1 .and. ictcount.eq.1 .and. kcall.eq.1 .and.
isearch.eq.0)
    .  call sbeamk(bmk,ek1)
c
c  loop over gauss points
c
  do 100 ii=1,ngp
    if(ngp.eq.4)eta=gauss4(ii)
    if(ngp.eq.5)eta=gauss5(ii)
    if(ngp.eq.7)eta=gauss7(ii)
    call shape(eta,ielem,aa)
c
c  multiply element displacement vector, eld (this is 'q'
c  in the thesis) by the shape function matrix, dsf, to get
c  the displacement gradient vector(d(s) in thesis, q here)
c
  do 20 kk=1,7
  20 q(kk)=0.0
c
  do 30 jj=1,7
    do 30 kk=1,9
  30 q(jj)=q(jj)+dsf(jj,kk)*eld(kk)
c
c  initialize bmn1, bmn2

```

```

c
do 35 kk=1,7
do 35 jj=1,7
bmn1(jj,kk)=0.0d0
35 bmn2(jj,kk)=0.0d0
c
c skip bmn1 and bmn2 comps first time through
c
if(icount .eq. 1 .and. ncount .eq. 1)goto 37
c
if(isarch.eq.1)call beamn1(q,bmn1,ek1,p1)
if(isarch.eq.1)call beamn2(q,bmn2,ek1,p1)
if(isarch.eq.0)call sbmn1(q,bmn1,ek1)
if(isarch.eq.0)call sbmn2(q,bmn2,ek1)
37   continue
c
c transpose the shape function matrix
c
do 40 jj=1,7
do 40 kk=1,9
40 dsftr(kk,jj)=dsf(jj,kk)
c
c create element independent incremental stiffness array,
c pkt, and element ind. equilibrium stiffness array, pkn
c
do 50 jj=1,7
do 50 kk=1,7
pkt(jj,kk)=bmk(jj,kk)+bmn1(jj,kk)+bmn2(jj,kk)
50 pkn(jj,kk)=bmk(jj,kk)+bmn1(jj,kk)/2.+bmn2(jj,kk)/3.
c
c post-multiply each array by the shape function matrix
c
do 60 jj=1,7
do 60 kk=1,9
pktd(jj,kk)=0.0
pknd(jj,kk)=0.0
do 60 ll=1,7
pktd(jj,kk)=pktd(jj,kk) + aa*pkt(jj,ll)*dsf(ll,kk)
60 pknd(jj,kk)=pknd(jj,kk) + aa*pkn(jj,ll)*dsf(ll,kk)
c
c Finally, pre-multiply these new arrays by the transpose
c of the shape function matrix to get the element incremental
c stiffness, stif, and element equilibrium stiffness, eln.
c Also multiply by the weighting factor for this particular
c gauss point. Note that these arrays are zeroed outside the
c loop over the gauss points since they accumulate (integrate)
c data over all the gauss points.
c
if/ngp.eq.4)wt=wt4(ii)
if/ngp.eq.5)wt=wt5(ii)
if/ngp.eq.7)wt=wt7(ii)
do 70 jj=1,9
do 70 kk=1,9
do 70 ll=1,7
stif(jj,kk)=stif(jj,kk)+wt*width*dsftr(jj,ll)*pktd(ll,kk)
eln(jj,kk)=eln(jj,kk)+wt*width*dsftr(jj,ll)*pknd(ll,kk)
70   continue
100  continue

```

```

c      write(6,1010)ielem
c      write(6,1005)
c      do 900 ii=1,9
c900  write(6,1000) (stif(ii,jj),jj=1,9)
1000  format(9(2x,d12.5))
1005  format(/, 'stif')
1010  format(i4)
      return
      end
c
c
c      subroutine shape(eta,ielem,aa)
c
c      implicit double precision (a-h,o-z)
c
c
c      common/shp/dsf(7,9)
c
c
c      common/input/tol,table(250),delem(250),vbound(2500),distld,
c      . vconc(2500),ey,enu,ht,e1,e2,g12,enu12,enu21,g13,g23,pthick,
c      . rad,linear,isotro,isarch,ishape,inctyp,ninc,imax,
c      . nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
c      . nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
c      . theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
c
c      initialize shape function matrix
c
c      do 10 ii=1,7
c      do 10 jj=1,9
10  dsf(ii,jj)=0.0
c
c      aa=(coord(ielem+1)-coord(ielem))*0.5
c
c      enter values into dsf
c      these include jacobian terms
c
c
c      Q1,Q3,Q2 and derivatives
c
c      dsf(1,1)=0.5*(eta**2-eta)
c      dsf(1,5)=1.0-eta**2
c      dsf(1,6)=0.5*(eta**2+eta)
c      dsf(2,1)=(eta-0.5)/aa
c      dsf(2,5)=-2.0*eta/aa
c      dsf(2,6)=(eta+0.5)/aa
c      dsf(3,3)=0.25*(2.0-3.0*eta+eta**3)
c      dsf(3,4)=0.25*aa*(1.0-eta-eta**2+eta**3)
c      dsf(3,8)=0.25*(2.0+3.0*eta-eta**3)
c      dsf(3,9)=0.25*aa*(-1.0-eta+eta**2+eta**3)
c      dsf(4,3)=0.25*(-3.0+3.0*eta**2)/aa
c      dsf(4,4)=0.25*(-1.0-2.0*eta+3.0*eta**2)
c      dsf(4,8)=0.25*(3.0-3.0*eta**2)/aa
c      dsf(4,9)=0.25*(-1.0+2.0*eta+3.0*eta**2)
c      dsf(5,3)=0.25*6.0*eta/aa**2
c      dsf(5,4)=0.25*(-2.0+6.0*eta)/aa
c      dsf(5,8)=-0.25*6.0*eta/aa**2

```

```

dsf(5,9)=0.25*(2.0+6.0*eta)/aa
dsf(6,2)=0.5*(1.0-eta)
dsf(6,7)=0.5*(1.0+eta)
dsf(7,2)=-0.5/aa
dsf(7,7)=0.5/aa

c
c temporary print
c
c do 100 ii=1,7
c 100 write(6,1000) (dsf(ii,jj),jj=1,9)
c 1000 format (9(2x,d12.5))
      return
      end

c
c
c subroutine postpr(icount,ncount,kcall,ndof)
c
c implicit double precision (a-h,o-z)
c
c character*64 gname
c
c
c common/chac/gname,fname
c
c common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
c
c common/input/tol,table(250),delem(250),vbound(2500),distld,
c . vconc(2500),ey,enu,ht,el,e2,g12,enu12,enu21,g13,g23,pthick,
c . rad,linear,isotro,isearch,ishape,inctyp,ninc,imax,
c . nelem,nbndry,nbound(250,5),ldtyp,nconc,iconc(2500),
c . nplies,nforc,iforc(2500),nstres,istres(250),ibndry(2500),
c . theta(20),idload(250),coord(251),width,nnod,pincr,eiter,ttpi
c
c common/stf/stif(9,9),elp(9),eln(9,9),eld(9)
c
c common/proc/gstif(2500,9),gn(2500,9),gf(2500),gd(2500),vperm(2500)
c
c dimension vforc(2500),xcoord(251),ycoord(251)
c pi=3.14159
c
c if ishape=1 print global x,y coords to file bshape
c if ishape=2 figure out symmetric coords as well
c
c     if (isearch.eq.0) then
c         go to 423
c     end if
c     if(ishape.eq.1.or.ishape.eq.2.and.ncount.eq.1.and.isearch.eq.1)then
c         open(8,file='bshape',status='new')
c         do 1 ii=1,nnod
c             if(ishape.eq.2)then
c                 xcoord(nnod-1+ii)=rad*cos(pi/2.0-coord(ii)/rad)
c                 xcoord(nnod+1-ii)=-rad*cos(pi/2.0-coord(ii)/rad)
c                 ycoord(nnod-1+ii)=rad*sin(pi/2.0-coord(ii)/rad)
c                 ycoord(nnod+1-ii)=ycoord(nnod-1+ii)
c             else

```

```

        xcoord(ii)=-rad*cos(pi/2.0+coord(ii)/rad-
coord(nnod)/(2*rad))
        ycoord(ii)=rad*sin(pi/2.0+coord(ii)/rad-coord(nnod)/(2*rad))
        endif
1      continue
do 100 ii=1,nnod
100    write(8,2000)xcoord(ii),ycoord(ii)
      if(ishape.eq.2)then
      do 110 ii=2,nnod
110    write(8,2000)xcoord(nnod+ii-1),ycoord(nnod+ii-1)
      endif
      write(8,2010)
      endif
c
c      global displacements for straight beams
c
423  if(ishape.eq.1.and.ncount.eq.1.and.isarch.eq.0)then
      open(8,file='bshape',status='new')
      do 2 ii=1,nnod
      xcoord(ii)= coord(nnod)-coord(ii)
      ycoord(ii)=0.0
2      write(8,2000)xcoord(ii),ycoord(ii)
      write(8,2010)
      endif
c
c      print out global displacements
c
      write(6,1000)
      write(6,1010)
      write(6,1020)ncount,icount
      write(6,1030)
      do 90 ii=0,nnod-1
c
c      x,y for arches
c
      if(ishape.eq.1.and.isarch.eq.1)then
      if(ishape.eq.1.and.isarch.eq.1.and.ii.eq.0)write(8,2020)ncount
          xcoord(ii+1)=-(rad-gd(5*ii+3))*cos(pi/2.0+coord(ii+1)/rad-coord(nnod)/(2*rad)-
          gd(5*ii+1)/rad)
          ycoord(ii+1)=(rad-gd(5*ii+3))*sin(pi/2.0+coord(ii+1)/rad-coord(nnod)/(2*rad)-
          gd(5*ii+1)/rad)
          write(8,2000)xcoord(ii+1),ycoord(ii+1)
      endif
      if(ishape.eq.2.and.isarch.eq.1)then
      if(ii.eq.0)write(8,2020)ncount
          xcoord(nnod+ii)=(rad-gd(5*ii+3))*cos(pi/2.0-coord(ii+1)/rad+gd(5*ii+1)/rad)
          xcoord(nnod-ii)=-xcoord(nnod+ii)
          ycoord(nnod+ii)=(rad-gd(5*ii+3))*sin(pi/2.0-coord(ii+1)/rad+gd(5*ii+1)/rad)
          ycoord(nnod-ii)=ycoord(nnod+ii)
      endif
c
c      x,y for straight beams
c
      if(ishape.eq.1.and.isarch.eq.0)then

```

```

        if(ishape.eq.1.and.isarch.eq.0.and.ii.eq.0)write(8,2020)ncount
          xcoord(ii+1)=coord(nnod)-coord(ii+1)-gd(5*ii+1)
          ycoord(ii+1)=-gd(5*ii+3)
          write(8,2000)xcoord(ii+1),ycoord(ii+1)
      endif
      write(6,1040)ii+1,(gd(5*ii+jj),jj=1,4)
90    write(6,1050)gd(5*ii+5)
      if(ishape.eq.2.and.isarch.eq.1)then
        do 95 ii=1,2*nnod-1
        write(8,2000)xcoord(ii),ycoord(ii)
      endif
      if(ishape.ge.1)write(8,2010)
c
c      compute equivalent forces requested
c
3      do 5 ii=1,ndof
      gf(ii)=0.0d0
      do 5 jj=1,9
      gstif(ii,jj)=0.0d0
5      gn(ii,jj)=0.0d0
c
c      loop over all elements for stiffness and forces
c
9      do 30 ielem=1,nelem
      do 10 ii=1,9
10      eld(ii)=gd(ii+(ielem-1)*5)
c
      call stiff(ielem,icount,ncount,kcal')
c
c      Assemble global stiffness array, gstif, global equilibrium
c      stiffness, gn, in banded form. Half-bandwidth=9. Also
c      assemble global force vector, gf.
c
      nr=(ielem-1)*5 + 1
      do 30 jj=0,8
      gf(nr+jj)=gf(nr+jj)+elp(jj+1)
      do 30 kk=1,9-jj
      gstif(nr+jj,kk)=gstif(nr+jj,kk)+stif(jj+1,kk+jj)
      if(linear.eq.1)goto 30
      gn(nr+jj,kk)=gn(nr+jj,kk)+eln(jj+1,kk+jj)
30      continue
c
c      calculate the residual force vector for nonlinear
c      analysis. -[gn]*{gd}+R=-[k+n1/2+n2/3]*{q}+R=gf
c
      do 60 jj=1,nforc
      ii=iforc(jj)
      add=0.
      do 50 kk=1,ii-1
      if(ii-kk+1 .gt. 9)goto 50
      add=add+gn(kk,ii-kk+1)*gd(kk)
50      continue
      res=0.
      do 55 ll=1,9
      if(ll+ii-1 .gt. ndof)goto 55
      res=res + gn(ii,ll)*gd(ll+ii-1)
55      continue
c

```

```

c      compute nodal force
c
c      vforc(jj)=res+add
60  continue
c
c      print nodal forces and create plot file
c
c      open(7,file='plot',status='new')
c      if(ncount.eq.1)write(7,*)0.0,0.0
c      write(6,1060)
c      do 70 ii=1,nforc
c         write(7,1065)gd(iforc(ii)),gd(iforc(ii)-2),vforc(ii)
70  write(6,1070)iforc(ii),vforc(ii)
1000 format(/)
1010 format(1x,'Results of nonlinear analysis')
1020 format(1x,'increment=',i3,' iteration=',i3)
1030 format(1x,'Node',7x,'V',13x,'Psi-s',13x,'W',13x,'W-s')
1040 format(1x,i4,4(2x,d12.5))
1050 format(1x,'Midnode v:',3x,d12.5)
1060 format(1x,/, 'Equivalent nodal forces:')
1065 format(1x,f12.5,2x,f12.5,2x,f12.5)
1070 format(1x,'DOF no:',i4,2x,'Force:',1x,d12.5)
2000 format(1x,f12.5,2x,f12.5)
2010      format(/)
2020 format(/,1x,i4)
      return
      end
c
c
c      subroutine beamk(bmk,ek1,p1)
c
c      implicit double precision (a-h,o-z)
c
c      common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
c      dimension bmk(7,7)
c      do 10 ii=1,7
c      do 10 jj=1,7
10  bmk(jj,ii)=0.0d0
c
c      bmk(2,2)=fe*p1**4-(2*de*p1**2)+ae
c
c      bmk(2,3)=de*p1**3-(ae*p1)
c
c      bmk(2,5)=- (he*ek1*p1**3)+fe*ek1*p1
c
c      bmk(2,7)=- (he*ek1*p1**3)+fe*(-p1**3+ek1*p1)+de*p1
c
c      bmk(3,3)=de*p1**4+ae*p1**2
c
c      bmk(3,5)=- (2*fe*ek1*p1**2)
c
c      bmk(3,7)=- (2*fe*ek1*p1**2)-(2*de*p1**2)
c
c      bmk(5,5)=ej*ek1**2*p1**2+he*ek1**2
c
c      bmk(5,7)=he*(ek1*p1**2+ek1**2)+ej*ek1**2*p1**2+fe*ek1
c

```

```

        bmk(7,7)=he*(2*ek1*p1**2+ek1**2)+fe*(p1**2+2*ek1)+ej*ek1**2
        . 2*p1**2+de
c
        bmk(4,4)=9*fs*ek1**2+6*ds*ek1+as
c
        bmk(4,6)=9*fs*ek1**2+6*ds*ek1+as
c
        bmk(6,6)=9*fs*ek1**2+6*ds*ek1+as
c
        do 100 ii=1,7
        do 100 jj=ii,7
100  bmk(jj,ii)=bmk(ii,jj)
        return
        end
c
c
c
        subroutine beamnl(q,bmn1,ek,p1)
c
c
c
        Note that k1 appears as 'ek' in this subroutine
c
c
        The equations in this subroutine were generated by MACSYMA.
c
c
        implicit double precision (a-h,o-z)
c
c
        common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
        dimension bmn1(7,7),q(7)
c
        bmn1(1,1)=p1**3*fe*(ek*(q(7)+q(5))-(p1**2*(q(7)+p1*(2*p1*q(3)-(3*q(3)
        . 2))))-(p1**5*ek*he*(q(7)+q(5)))+p1**3*de*(q(7)+p1*(3*p1*q(3
        . )-(4*q(2))))+p1**2*ae*(-(p1*q(3))+q(2))
        bmn1(1,2)=p1**3*fe*(ek*q(6)-(p1**2*q(6))+ek*q(4)+2*p1**2*q(4)+3*p1**3*q(1)
        . -(p1**5*ek*he*(q(6)+q(4)))+p1**3*de*(q(6)-(3*q(4))-
        . 4*p1*q(1)))+p1*ae*(q(4)+p1*q(1))
        bmn1(1,3)=-(2*p1**4*fe*(ek*(q(6)+q(4))-(p1**2*(q(6)-(p1*q(1))))))-
        . (p1**4*de*(2*p1*(q(6)-q(4)-(3*p1*q(1))))+2*p1**6*ek*he*(q(6)+q(4
        . ))-(p1**2*ae*(q(4)+p1*q(1))))
        bmn1(1,4)=p1**2*fe*(3*ek*q(7)-(2*p1**2*q(7))+2*ek*q(5)-(2*p1**2*ek*q
        . (3))+p1*ek*q(2)+2*p1**3*q(2))+p1**2*ek*he*(ek*q(7)-(3*p1**2*q(7))+ek*q
        . (5)-(2*p1**2*q(5))+2*p1**4*q(3)-(p1**3*q(2)))+p1**2*de*(2*q(7)+p1*(p1*q(3)-(3*q(2)))-(p1**4*ek**2*ej*(q(7)+q(5)))+p1*ae*(-(p1*q(3))+q(2))
        bmn1(1,5)=p1**2*ek**2*(ek*q(6)-(p1**2*q(6))+ek*q(4)-(2*p1**2*q(4))-
        . p1**3*q(1))+p1**2*ek*fe*(q(6)+2*q(4)+p1*q(1))-(p1**4*ek**2*ej*(q(6)+q(4)))
        bmn1(1,6)=p1**2*fe*(2*ek*q(7)-(p1**2*q(7))+ek*q(5)-(2*p1**2*ek*q(3))
        . +2*p1**4*q(3)+p1*ek*q(2)-(p1**3*q(2)))+p1**2*ek*he*(ek*q(7)-(2*p1**2*q
        . (7))+ek*q(5)-(p1**2*q(5))+2*p1**4*q(3)-(p1**3*q(2)))-(p1**4*ek**2*ej*(q(7)+q(5)))+p1**2*de*(q(7)-(p1*(2*p1*q(3)-q(2))))
        bmn1(1,7)=p1**2*fe*(2*ek*q(6)-(p1**2*q(6))+3*ek*q(4)-(2*p1**2*q(4))
        . +p1*ek*q(1)-(p1**3*q(1))+p1**2*ek*he*(ek*q(6)-(2*p1**2*q(6))+ek*q(4
        . -(3*p1**2*q(4))-(p1**3*q(1)))+p1**2*de*(q(6)+2*q(4)+p1*q(1))-(p1
        . **4*ek**2*ej*(q(6)+q(4)))
        bmn1(2,2)=-(p1*fe*(3*p1**2*(q(7)+p1*(2*p1*q(3)-(3*q(2))))+2*p1**2*q(6)**2*q(7)-(3*ek*q(7)-(3*ek*q(5))))+p1*de*((2*q(6)**2+3)*q(7)+3*p1*(3*p1*q(3)-(4*q(2))))-(3*p1**3*ek*he*(q(7)+q(5)))-(3*ae*(p1*q(3)-q(2)))
        bmn1(2,3)=-(6*p1**2*fe*(ek*(q(7)+q(5))-(p1**2*(q(7)-(p1*q(2))))))-
        . (p1**2*de*(2*(q(6)**2+3)*q(7)+3*p1*(p1*q(3)-(3*q(2)))))+6*p1**4*ek*he*(q(7)+q(5))+3*p1*ae*(p1*q(3)-q(2))
        bmn1(2,4)=p1**2*fe*(3*ek*q(6)-(2*p1**2*q(6))+4*ek*q(4)+p1*ek*q(1)+2*p1**3*q(1)+p1**2*ek*he*(ek*q(6)-(3*p1**2*q(6))+ek*q(4)-(4*p1**2*q(4)))-(p1**3*q(1))+p1**2*de*(2*q(6)-q(4)-(3*p1*q(1)))-(p1**4*ek**2*ej*(q(6)+q(4)))+ae*(q(4)+p1*q(1))
        bmn1(2,5)=3*ek*he*(ek*(q(7)+q(5))-(p1**2*(q(7)-(p1*(2*p1*q(3)-q(2))))+2*p1*de*(2*q(6)-q(4)-(3*p1*q(1)))-(p1**4*ek**2*ej*(q(6)+q(4)))+ae*(q(4)+p1*q(1)))

```

```

. )))) +ek*fe*((2*q(6)**2+3)*q(7) - (3*p1*(2*p1*q(3) - q(2)))) - (3*p1**2*ek
. **2*ej*(q(7) + q(5)))
bmn1(2, 6) = -(p1**2*fe*(p1*q(6)*q(7) - (2*ek*q(6)) + p1**2*q(6) - (3*ek*q(4))
. + 2*p1**2*q(4) - (p1*ek*q(1)) + p1**3*q(1)) + p1*de*(q(6)*q(7) + p1*(q(6) + 2*
. q(4) + p1*q(1))) + p1**2*ek*he*(ek*q(6) - (2*p1**2*q(6)) + ek*q(4) - (3*p1**2*
. q(4) - (p1**3*q(1))) - (p1**4*ek**2*ej*(q(6) + q(4)))
bmn1(2, 7) = -da*((4*q(6)**2+3)*q(7) + p1*(q(6)*q(6) - (2*p1*q(6)**2*q(3))) - (6
. *p1*q(3)) + (2*q(6)**2+3)*q(2))) + fe*(4*ek*q(6)**2*q(7) + 6*ek*q(7) - (3*p1
. **2*q(7)) - (p1**3*q(6)*q(6)) + 2*ek*q(6)**2*q(5) + 3*ek*q(5) - (6*p1**2*ek*q(3)) +
. 6*p1**4*q(3) - (2*p1**3*q(6)**2*q(2)) + 3*p1*ek*q(2) - (3*p1**3*q(2)) + 3*ek*he
. *(ek*q(7) - (2*p1**2*q(7)) + ek*q(5) - (p1**2*q(5)) + 2*p1**4*q(3) - (p1**3*q(2)
. 2)) - (3*p1**2*ek**2*ej*(q(7) + q(5)))
bmn1(3, 3) = p1**3*de*((2*q(6)**2+9)*q(7) - (3*p1*(2*p1*q(3) + q(2)))) + 9*
. p1**3*ek*fe*(q(7) + q(5)) - (3*p1**2*ae*(p1*q(3) - q(2)))
bmn1(3, 4) = -(p1**3*ek*he*(ek*(q(6) + q(4)) + 2*p1**2*(q(6) - (p1*q(1)))) -
. (p1**3*de*(3*q(6) + 2*q(4) - (p1*q(1)))) - (2*p1**3*ek*fe*(2*q(6) +
. 3*q(4) + p1*q(1))) - (2*p1**5*ek**2*ej*(q(6) + q(4))) + p1*ae*(-q(4) -
. (p1*q(1)))
bmn1(3, 5) = -(p1*ek*he*(6*p1**2*(q(7) - (p1*q(2))) + 2*p1**2*q(6)**2*q(7) + 3
. *ek*q(7) + 3*ek*q(5)) - (p1*ek*fe*((2*q(6)**2+3)*q(7) - (3*p1*(3*p1*q(3) - (2*
. 2*q(2)))) - (6*p1**3*ek**2*ej*(q(7) + q(5)))
bmn1(3, 6) = -(p1**2*de*(2*q(6)*q(7) + p1*(q(6) + 3*q(4) + 2*p1*q(1))) - (p1
. **3*ek*he*(ek*q(6) + 4*p1**2*q(6) + ek*q(4) + 2*p1**2*q(4) - (2*p1**3*q(1))
. ) - (2*p1**3*fe*(ek*q(6) + p1**2*q(6) + 2*ek*q(4) + p1*ek*q(1) - (p1**3*q(1))
. ) - (2*p1**5*ek**2*ej*(q(6) + q(4)))
bmn1(3, 7) = -(p1*de*((4*q(6)**2+3)*q(7) + p1*(2*q(6)*q(6) - (2*p1*q(6)**2*q(
. 3)) - (9*p1*q(3)) + 2*(q(6)**2+3)*q(2))) - (p1*fe*(4*ek*q(6)**2*q(7) + 6*ek*
. q(7) + 6*p1**2*q(7) + 2*ek*q(6)**2*q(5) + 3*ek*q(5) - (9*p1**2*ek*q(3)) + 6*p1*ek*q(
. 2) - (6*p1**3*q(2))) - (p1*ek*he*(3*ek*q(7) + 12*p1**2*q(7) + 2*p1**2*q(6)**2
. *2*q(5) + 3*ek*q(5) + 6*p1**2*q(5) - (6*p1**3*q(2))) - (6*p1**3*ek**2*ej*(q(7) + q(
. 5)))
bmn1(4, 4) = -p1*ek*he*(2*ek*q(7) + 4*p1**2*q(7) + 2*ek*q(5) - (p1**2*ek*q(3)) +
. p1*ek*q(2) - (4*p1**3*q(2))) + p1**3*ek**2*ej*(7*q(7) + 4*q(5) - (p1*(2*p1*q
. (3) + q(2)))) + p1*ek*fe*(5*q(7) + 3*q(5) - (2*p1*(3*p1*q(3) - (2*q(2)))) + p1
. *de*(3*q(7) - (p1*(2*p1*q(3) + q(2)))) + 3*p1**3*ek**3*el*(q(7) + q(5)) + ae*(-(p1*q(3) + q(2)))
bmn1(4, 5) = -p1*ek*he*(ek*q(6) + 2*p1**2*q(6) + 2*ek*q(4) + p1*ek*q(1) - (2*p1**3
. *q(1)) + p1**3*ek**2*ej*(5*q(6) + 4*q(4) - (p1*q(1))) + p1*ek*fe*(q(6) + 3*q(4) + 2*p1*q(1)) + 3*p1**3*ek**3*el*(q(6) + q(4))
bmn1(4, 6) = -p1*fe*(2*ek*q(7) + 2*p1**2*q(7) + ek*q(5) - (4*p1**2*ek*q(3)) + 3*
. p1*ek*q(2) - (2*p1**3*q(2)) + p1*ek*he*(ek*q(7) + 7*p1**2*q(7) + ek*q(5) + 2*p1
. **2*q(5) - (p1**2*ek*q(3)) - (2*p1**4*q(3)) + p1*ek*q(2) - (3*p1**3*q(2)) + p1**3
. *ek**2*ej*(8*q(7) + 5*q(5) - (p1*(2*p1*q(3) + q(2)))) + 3*p1**3*ek**3*el
. *(q(7) + q(5)) + p1*de*(q(7) - (p1*(3*p1*q(3) - (2*q(2)))) + p1*ek*he*(2*ek*q(6) + 2*p1**2*q(6) + 5*ek*q(4) + 3*p1*ek*q(1) - (2*p1
. **3*q(1)) + p1*ek*he*(ek*q(6) + 7*p1**2*q(6) + 2*ek*q(4) + 4*p1**2*q(4) + p1*
. ek*q(1) - (3*p1**3*q(1)) + p1**3*ek**2*ej*(8*q(6) + 7*q(4) - (p1*q(1))) + p1
. *de*(q(6) + 3*q(4) + 2*p1*q(1)) + 3*p1**3*ek**3*el*(q(6) + q(4))
bmn1(5, 5) = -3*p1*ek**2*ej*((2*q(6)**2+3)*q(7) - (p1*(2*p1*q(3) + q(2)))) + 9
. *p1*ek**3*el*(q(7) + q(5)) - (3*ek**2*he*(p1*q(3) - q(2)))
bmn1(5, 6) = -p1*ek*he*(p1*q(6)*q(7) + 3*p1**2*q(6) + ek*q(4) + 2*p1**2*q(4) + p1*ek
. *q(1) - (p1**3*q(1)) + ek*fe*(q(6)*q(7) + p1*(q(4) + p1*q(1))) + p1**3*ek**2*ej
. *(6*q(6) + 5*q(4) - (p1*q(1))) + 3*p1**3*ek**3*el*(q(6) + q(4))
bmn1(5, 7) = -p1*ek**2*ej*(2*(4*q(6)**2+9)*q(7) + 3*(2*q(6)**2+3)*q(5) - (3*
. p1*(2*p1*q(3) + q(2))) + ek*he*(8*p1*q(6)**2*q(7) + 9*p1*q(7) + p1**2*q(6)*q(6
. ) - (2*p1**3*q(6)**2*q(3)) - (3*p1*ek*q(3)) - (6*p1**3*q(3)) + 3*ek*q(2) - (3*p1**2
. *q(2)) + 9*p1*ek**3*el*(q(7) + q(5)) + ek*fe*(q(6)*q(6) - (2*p1*q(6)**2*q(3)) - (3*p1*q(3) + q(2)) + 2*(q(6)**2+3)*q(2))
bmn1(6, 6) = -p1**2*ek*he*(9*p1*q(7) + 3*p1*q(5) - (p1*ek*q(3)) - (4*p1**3*q(3)
. ) + ek*q(2) - (2*p1**2*q(2)) + p1**2*fe*(3*p1*q(7) - (2*p1*ek*q(3)) - (2*p1**3
. *q(3)) + 2*ek*q(2) - (p1**2*q(2)) + p1**3*ek**2*ej*(9*q(7) + 6*q(5) - (p1*
. (2*p1*q(3) + q(2)))) + 3*p1**3*ek**3*el*(q(7) + q(5)) + p1**2*de*(-(p1*q(3)) + q(2)))
bmn1(6, 7) = -fe*(2*ek*q(6)*q(7) + 2*p1**2*q(6)*q(7) + 3*p1**3*q(6) + ek*q(6)*q(5)
. + 2*p1*ek*q(4) + 2*p1**3*q(4) - (p1**3*q(6)*q(2)) + 2*p1**2*ek*q(1) - (p1**4*q(1))
. + p1*ek*he*(2*p1*q(6)*q(7) + 9*p1**2*q(6) + p1*q(6)*q(5) + ek*q(4) + 7*p1**2*q(4)
. + p1*ek*q(1) - (2*p1**3*q(1))) + de*(2*p1*q(6)*q(7) + p1*q(4) - (2*p1**2*q(6)*q(3
. )) + p1*q(6)*q(2) + p1**2*q(1)) + p1**3*ek**2*ej*(9*q(6) + 8*q(4) - (p1*q(1)))
. + 3*p1**3*ek**3*el*(q(6) + q(4))
bmn1(7, 7) = -p1*ek**2*ej*(3*(2*q(6)**2+9)*q(7) + 2*(4*q(6)**2+9)*q(5) - (3*
. p1*(2*p1*q(3) + q(2))) + ek*he*(12*p1*q(6)**2*q(7) + 27*p1*q(7) + 2*p1**2*q(6)
. *q(6) + 8*p1*q(6)**2*q(5) + 9*p1*q(5) - (3*p1*ek*q(3)) - (12*p1**3*q(3)) + 3*ek*q(2

```

```

. )-(6*p1**2*q(2)))+fe*(6*p1*q(6)**2*q(7)+9*p1*q(7)+2*ek*q(6)*q(6)+2*p1
. **2*q(6)*q(6)-(4*p1*ek*q(6)**2*q(3))-(6*p1*ek*q(3))-(6*p1**3*q(3))+4*ek*q(6)
. **2*q(2)+6*ek*q(2)-(3*p1**2*q(2)))+9*p1*ek**3*el*(q(7)+q(5))+de
. *(2*q(6)*q(6)-(4*p1*q(6)**2*q(3))-(3*p1*q(3))+(4*q(6)**2+3)*q(2))
c
c      do 100 ii=1,7
c      do 100 jj=ii,7
100  bmn1(jj,ii)=bmn1(ii,jj)
      return
      end
c
c
c      subroutine bmn2(q,bmn2,ek,p1)
c
c      Note that k1 appears as 'ek' in this subroutine
c
c      The equations in this subroutine were generated by MACSYMA.
c
c      implicit double precision (a-h,o-z)
c
c      common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
c      dimension bmn2(7,7),q(7)
c
T0=4.0/3.0*p1**6*ek**2*el*((q(7)+q(5))**2+4*p1**2*(q(6)+q(4))**2)
. -(1.0/6.0*p1**2*he*(12*p1**2*ek*q(6)**2*q(7)**2-(ek**2*q(7)**2)+12*
. p1**2*ek*q(7)**2-(12*p1**4*q(7)**2)+12*p1**2*ek*q(6)**2*q(5)*q(7)-(2*ek**2*q(5)*q(7))+12*p1**2*ek*q(5)*q(7)-(12*p1**4*ek*q(3)*q(7))+24*p1**5*q(2)*q(7)-(7*p1**2*ek**2*q(6)**2)+44*p1**4*ek*q(6)**2-(36*p1**6*q(6)**2)-(14*p1**2*ek**2*q(4)*q(6))+88*p1**4*ek*q(4)*q(6)+72*p1**7*q(1)*q(6)-(ek**2*q(5)**2)-(12*p1**4*ek*q(3)*q(5))-(7*p1**2*ek**2*q(4)**2)+44*p1**4*ek*q(4)**2-(12*p1**6*q(2)**2)-(36*p1**8*q(1)**2)))
T0=T0+p1**2*fe*(2*ek*q(6)**2*q(7)**2-(18*p1**2*q(6)**2*q(7)**2)+2*ek*q(7)**2-(9*p1**2*q(7)**2)-(10*p1**3*q(6)*q(6)*q(7))+2*ek*q(6)**2*q(5)*q(7)+2*ek*q(5)*q(7)-(8*p1**4*q(6)**2*q(3)*q(7))-(4*p1**2*ek*q(3)*q(7))+28*p1**4*q(3)*q(7)-(10*p1**3*q(6)**2*q(2)*q(7))+2*p1*ek*q(2)*q(7)-(10*p1**3*q(2)*q(7))+10*p1**2*ek*q(6)**2-(19*p1**4*q(6)**2)+26*p1**2*ek*q(4)*q(6)-(68*p1**4*q(4)*q(6))+6*p1**3*ek*q(1)*q(6)-(30*p1**5*q(1)*q(6))-(4*p1**2*ek*q(3)*q(5))+2*p1*ek*q(2)*q(5)+16*p1**2*ek*q(4)**2+32*p1**4*q(4)**2+6*p1**3*ek*q(1)*q(4)+132*p1**5*q(1)*q(4)+8*p1**6*q(3)**2-(44*p1**5*q(2)*q(3))+27*p1**4*q(2)**2+81*p1**6*q(1)**2)/6.0
bmn2(1,1)=T0+p1**2*de*(2*q(6)**2*q(7)**2+2*p1*q(6)*q(6)*q(7)
. )-(4*p1**2*q(3)*q(7))+2*p1*q(6)**2*q(2)*q(7)+2*p1*q(2)*q(7)+7*p1**2*q(6)**2+20*p1**2*q(4)*q(6)+6*p1**3*q(1)*q(6)-(5*p1**2*q(4)**2)-(30*p1**3*q(1)*q(4))-(3*p1**4*q(3)**2)+10*p1**3*q(2)*q(3)-(6*p1**2*q(2)**2)-(18*p1**4*q(1)**2)/6.0-(1.0/3.0*p1**4*ek*ej*(3*ek*q(7)**2-(8*p1**2*q(7)**2)-(8*p1**2*q(6)*q(7))+6*ek*q(5)*q(7)-(8*p1**2*q(5)*q(7)))+8*p1**3*q(2)*q(7)+8*p1**2*ek*q(6)**2-(24*p1**4*q(6)**2)+16*p1**2*ek*q(4)*q(6)-(24*p1**4*q(4)*q(6))+24*p1**5*q(1)*q(6)+3*ek*q(5)**2+8*p1**3*q(2)*q(5)+8*p1**2*ek*q(4)**2+24*p1**5*q(1)*q(4)))+p1**2*ae*(3*q(4)**2+6*p1*q(1)*q(4)+p1**2*q(3)**2-(2*p1*q(2)*q(3))+q(2)**2+3*p1**2*q(1)**2)/2.0
T0=-(1.0/3.0*p1**2*he*(2*p1**2*ek*q(6)**2*q(6)*q(7)-(3*ek**2*q(6)*q(7))+16*p1**2*ek*q(6)*q(7)-(12*p1**4*q(6)*q(7))+2*p1**2*ek*q(6)**2*q(4)*q(7)-(3*ek**2*q(5)*q(6))+8*p1**2*ek*q(5)*q(6)-(8*p1**4*ek*q(3)*q(6))+12*p1**5*q(1)*q(7)-(3*ek*2*q(6)-3*ek**2*q(4)*q(5))+8*p1**2*ek*q(4)*q(5)-(8*p1**4*ek*q(3)*q(4)))-(12*p1**6*q(1)*q(2)))
T0=T0+p1**2*fe*(2*ek*q(6)**2*q(7)-(p1**2*q(6)**2*q(6)*q(7))+4*ek*q(6)*q(7)-(5*p1**2*q(6)*q(7))+2*ek*q(6)**2*q(4)*q(7)-(6*p1**2*q(6)**2*q(4)*q(7))+5*ek*q(4)*q(7)-(10*p1**2*q(4)*q(7))-(5*p1**3*q(6)**2*q(1)*q(7))+p1*ek*q(1)*q(7)-(5*p1**3*q(1)*q(7))+2*ek*q(5)*q(6)-(3*p1**2*ek*q(3)*q(6))+p1**4*q(6)+10*p1**4*q(3)*q(6)+p1*ek*q(2)*q(6)-(5*p1**3*q(2)*q(6))+3*ek*q(4)*q(5)+p1*ek*q(1)*q(5)-(3*p1**2*ek*q(3)*q(4))-(12*p1**4*q(3)*q(4))+p1*ek*q(2)*q(4)+22*p1**3*q(2)*q(4)-(22*p1**5*q(1)*q(3))+27*p1**4*q(1)*q(2))/3.0+p1**2*de*(3*q(6)**2*q(7)+3*q(6)*q(7)+4*q(6)**2*q(4)*q(7)+4*q(4)*q(7)+p1*q(6)**2*q(1)*q(7)+p1*q(1)*q(7)-(4*p1**2*q(3)*q(6))+p1*q(2)*q(6)+p1**2*q(3)*q(4)-(5*p1*q(2)*q(4))+5*p1**3*q(1)*q(3)-(6*p1**2*q(1)*q(2)))/3.0
bmn2(1,2)=T0-(1.0/3.0*p1**4*ek*ej*(5*ek*q(6)*q(7)-(16*p1**2*q(6)*q(7))+5*ek*q(4)*q(7)-(8*p1**2*q(4)*q(7))+8*p1**3*q(1)*q(7)+5*ek*q(5)*q(6)-(8*p1**2*q(5)*q(6))+8*p1**3*q(2)*q(6)+5*ek*q(4)*q(5)+8*p1**3*q(1)

```

```

. *q(5)+8*p1**3*q(2)*q(4)))+4*p1**6*ek**2*el*(q(6)+q(4))*(q(7)+q(5)
. )-(p1*ae*(p1*q(3)-q(2))*(q(4)+p1*q(1)))
T0=-(1.0/3.0*p1**3*fe*(2*ek*q(6)**2*q(6)*q(7)-(4*p1**2*q(6)**2*q(6)*q
. (7))+4*ek*q(6)*q(7)+2*p1**2*q(6)*q(7)+2*ek*q(6)**2*q(4)*q(7)+6*ek*q(4)*
. q(7)-(12*p1**2*q(4)*q(7))+4*p1**3*q(6)**2*q(1)*q(7)+2*p1*ek*q(1)*q(7)-(4
. 14*p1**3*q(1)*q(7))+2*ek*q(5)*q(6)-(5*p1**2*ek*q(3)*q(6))+8*p1**4*q(3)
. *q(6)+3*p1*ek*q(2)*q(6)-(10*p1**3*q(2)*q(6))+4*ek*q(4)*q(5)+2*p1*ek*q(1)
. )*q(5)-(5*p1**2*ek*q(3)*q(4))+3*p1*ek*q(2)*q(4)+12*p1**3*q(2)*q(4)-(8*
. p1**5*q(1)*q(3))+22*p1**4*q(1)*q(2))+p1**3*ek*he*(4*p1**2*q(6)**2*q
. (6)*q(7)-(3*ek*q(6)*q(7))+2*p1**2*q(6)*q(7)+4*p1**2*q(6)**2*q(4)*q(7)-
. (3*ek*q(4)*q(7))+8*p1**2*q(4)*q(7)+6*p1**3*q(1)*q(7)-(3*ek*q(5)*q(6))
. +2*p1**2*q(5)*q(6)-(8*p1**4*q(3)*q(6))+8*p1**3*q(2)*q(6)-(3*ek*q(4)*q
. (5))+8*p1**2*q(4)*q(5)+6*p1**3*q(1)*q(5)-(8*p1**4*q(3)*q(4))+8*p1**3*
. q(2)*q(4))/3.0
bnn2(1,3)=T0-(1.0/3.0*p1**3*de*(3*q(6)**2*q(6)*q(7)+3*q(6)*q(7)-
. 3*q(6)**2*q(4)*q(7)+5*q(4)*q(7)+2*p1*q(1)*q(7)-(7*p1**2*q(3)*q(6))+4*
. p1*q(2)*q(6)-(4*p1**2*q(3)*q(4))-(p1*q(2)*q(4))+3*p1**3*q(1)*q(3)-(5*
. p1**2*q(1)*q(2)))-(2.0/3.0*p1**5*ek**2*ej*(q(6)+q(4))*(q(7)+q(5)
. ))+p1**2*ae*(p1*q(3)-q(2))*(q(4)+p1*q(1))
T0=-(4.0/3.0*p1**5*ek**3*re*((q(7)+q(5))**2+3*p1**2*(q(6)+q(4))**2
. 2)+p1**3*ek*ej*(3*ek*q(6)**2*q(7)**2+7*ek*q(7)**2-(4*p1**2*q(7)**2)
. +7*ek*q(6)**2*q(5)*q(7)+7*ek*q(5)*q(7)-(2*p1**2*ek*q(3)*q(7))-(5*p1*ek*q
. 2)*q(7))+8*p1**3*q(2)*q(7)+22*p1**2*ek*q(6)**2-(12*p1**4*q(6)**2)+28*
. p1**2*ek*q(4)*q(6)-(16*p1**3*ek*q(1)*q(6))+24*p1**5*q(1)*q(6)-(2*p1**2*
. ek*q(3)*q(5))-(5*p1*ek*q(2)*q(5))+6*p1**2*ek*q(4)**2-(16*p1**3*ek*q(1)*q
. (4))-(4*p1**4*q(2)**2)-(12*p1**6*q(1)**2))/3.0
T0=T0+p1*fe*(2*ek*q(6)**2*q(7)**2-(12*p1**2*q(6)**2*q(7)**2)+2*ek*q(7)
. **2-(2*p1**2*q(7)**2)+4*p1*ek*q(6)*q(6)*q(7)-(12*p1**3*q(6)*q(6)*q(7))+2*
. ek*q(6)**2*q(5)*q(7)+2*ek*q(5)*q(7)-(4*p1**2*ek*q(6)**2*q(3)*q(7))-(12*p1
. **2*ek*q(3)*q(7))+24*p1**4*q(3)*q(7)+4*p1*ek*q(6)**2*q(2)*q(7)-(12*p1**3
. *q(6)**2*q(2)*q(7))+10*p1*ek*q(2)*q(7)-(20*p1**3*q(2)*q(7))+10*p1**2*ek*
. q(6)**2+2*p1**4*q(6)**2+4*p1**2*ek*q(4)*q(6)-(64*p1**4*q(4)*q(6))+26
. *p1**3*ek*(q(1)*q(6))-(68*p1**5*q(1)*q(6))-(8*p1**2*ek*q(3)*q(5))+6*p1*ek*
. q(2)*q(5)+39*p1**2*ek*q(4)**2+32*p1**3*ek*q(1)*q(4)+64*p1**5*q(1)*q(4)
. +5*p1**4*ek*q(3)**2-(6*p1**3*ek*q(2)*q(3))-(24*p1**5*q(2)*q(3))+p1**2*ek*
. q(2)**2+22*p1**4*q(2)**2+3*p1**4*ek*q(1)**2+66*p1**6*q(1)**2)/6.0
T0=T0-(1.0/6.0*p1*ek*he*(4*p1**2*q(6)**2*q(7)**2-(ek*q(7)**2)-(8*p1**2
. 2*q(7)**2)+4*p1**3*q(6)*q(6)*q(7)-(2*ek*q(5)*q(7))-(8*p1**4*q(6)**2*q(3)
. *q(7))+6*p1**2*ek*q(3)*q(7)-(16*p1**4*q(3)*q(7))+4*p1**3*q(6)**2*q(2)*q
. (7)-(6*p1*ek*q(2)*q(7))+32*p1**3*q(2)*q(7)-(7*p1**2*ek*q(6)**2)-(16*p1
. **4*q(6)**2)-(28*p1**2*ek*q(4)*q(6))+56*p1**4*q(4)*q(6)-(14*p1**3*ek*q
. (1)*q(6))+88*p1**5*q(1)*q(6)-(ek*q(5)**2)+6*p1**2*ek*q(3)*q(5)-(16*p1
. **4*q(3)*q(5))-(6*p1*ek*q(2)*q(5))+16*p1**3*q(2)*q(5)-(21*p1**2*ek*q(4)
. )**2)+72*p1**4*q(4)**2-(14*p1**3*ek*q(1)*q(4))+88*p1**5*q(1)*q(4)+8*p1
. **6*q(3)**2-(16*p1**5*q(2)*q(3)))
bnn2(1,4)=T0+p1*de*(2*q(6)**2*q(7)**2+8*p1*q(6)*q(6)*q(7)-(6*p1**2*q(6)**2*q(3)*q(7))-(10*p1**2*q(3)*q(7))+8*p1*q(6)**2*q(2)*q(7)+8
. *p1*q(2)*q(7)+7*p1**2*q(6)**2+34*p1**2*q(4)*q(6)+20*p1**3*q(1)*q(6)+
. 12*p1**2*q(4)**2-(10*p1**3*q(1)*q(4))+4*p1**4*q(3)**2+2*p1**3*q(2)*q(4
. )-(5*p1**2*q(2)**2)-(15*p1**4*q(1)**2))/6.0+6*p1**3*ek**2*el*(3*ek*
. q(7)**2-(12*p1**2*q(7)**2)-(12*p1**2*q(6)**2+2*q(5)*q(7))+6*ek*q(5)*q(7)
. -(12*p1**2*q(5)*q(7))+12*p1**3*q(2)*q(7)+9*p1**2*ek*q(6)**2-(32*p1**4*
. q(6)**2)+18*p1**2*ek*q(4)*q(6)-(32*p1**4*q(4)*q(6))+32*p1**5*q(1)*q(6)
. )+3*ek*q(5)**2+12*p1**3*q(2)*q(5)+9*p1**2*ek*q(4)**2+32*p1**5*q(1)*q(4)
. )/3.0+p1*ae*(3*q(4)**2+6*p1*q(1)*q(4)+p1**2*q(3)**2-(2*p1*q(2)*q
. (3))+q(2)**2+3*p1**2*q(1)**2)/2.0
T0=p1**3*ek*ej*(7*ek*q(6)**2*q(6)*q(7)-(8*p1**2*q(6)**2*q(6)*q(7))+13*
. ek*q(6)*q(7)-(8*p1**2*q(6)*q(7))+7*ek*q(6)**2*q(4)*q(7)+7*ek*q(4)*q(7)+
. 8*p1**3*q(6)**2*q(1)*q(7)-(6*p1*ek*q(1)*q(7))+8*p1**3*q(1)*q(7)+6*ek*q(5)
. )*q(6)-(2*p1**2*ek*q(3)*q(6))-(5*p1*ek*q(2)*q(6))+8*p1**3*q(2)*q(6)-(6
. *p1*ek*q(1)*q(5))-(2*p1**2*ek*q(3)*q(4))-(5*p1*ek*q(2)*q(4))-(8*p1**4*q
. (1)*q(2))/3.0-(2.0/3.0*p1**3*q(6)**2*q(6)*q(7)+6*p1**2*q(6)*q(7)-ek*
. 3*ek*q(6)*q(7))+10*p1**2*q(6)*q(7)+6*p1**2*q(6)**2*q(4)*q(7)-(3*ek*q(4)
. *q(7))+6*p1**2*q(4)*q(7)-(4*p1**3*q(1)*q(7))-(3*ek*q(5)*q(6))+4*p1**2
. *q(5)*q(6)-(6*p1**3*q(2)*q(6))-(3*ek*q(4)*q(5))-(4*p1**3*q(1)*q(5))-
. (6*p1**3*q(2)*q(4)))
bnn2(1,5)=T0+p1*ek*he*(6*p1**2*q(6)**2*q(6)*q(7)+6*p1**2*q(6)*q(7)+ek
. *q(4)*q(7)-(6*p1**3*q(6)**2*q(1)*q(7))+p1*ek*q(1)*q(7)-(6*p1**3*q(1)*q
. 7)-(3*p1**2*ek*q(3)*q(6))+2*p1**4*q(3)*q(6)+3*p1*ek*q(2)*q(6)-(8*p1**3
. *q(2)*q(6))+ek*q(4)*q(5)+p1*ek*q(1)*q(5)-(3*p1**2*ek*q(3)*q(4))+8*p1**4
. *q(3)*q(4)+3*p1*ek*q(2)*q(4)-(8*p1**3*q(2)*q(4))+6*p1**5*q(1)*q(3)/
. 3.0+p1*ek*fe*(q(6)**2*q(4)*q(7)+q(4)*q(7)+p1*q(6)**2*q(1)*q(7)+p1*q(1)

```

```

. )*q(7)-(2*p1**2*q(3)*q(6))+2*p1*q(2)*q(6)-(4*p1**2*q(3)*q(4))+3*p1*q(
. 2)*q(4)-(2*p1**3*q(1)*q(3))+p1**2*q(1)*q(2)/3.0-(8.0/3.0*p1**5*ek**3
. *re*(q(6)+q(4))*(q(7)+q(5)))
T0=-(4.0/3.0*p1**5*ek**3*re*((q(7)+q(5))**2+3*p1**2*(q(6)+q(4))**2))
. 2))
T0=T0+p1**2*he*(4*p1*ek*q(6)**2*q(7)**2+10*p1*ek*q(7)**2-(6*p1**3*q(7)
. **2)-(4*p1**2*ek*q(6)*q(6)*q(7))+6*p1*ek*q(6)**2*q(5)*q(7)+6*p1*ek*q(5)*q(7
. )-(2*p1**2*ek*q(6)*q(4)*q(7))+4*p1**3*ek*q(6)**2*q(3)*q(7)-(3*p1*ek**2*q(3)
. *q(7))+2*p1**3*ek*q(3)*q(7)-(2*p1**2*ek*q(6)**2*q(2)*q(7))+3*ek**2*q(2)*
. q(7)-(16*p1**2*ek*q(2)*q(7))+12*p1**4*q(2)*q(7)+30*p1**3*ek*q(6)**2-(1
. 8*p1**5*q(6)**2)+7*p1**4*ek**2*q(4)*q(6)+16*p1**3*ek*q(4)*q(6)+7*p1**2*ek
. **2*q(1)*q(6)-(44*p1**4*ek*q(1)*q(6))+36*p1**6*q(1)*q(6)-(3*p1*ek**2*q
. (3)*q(5))+2*p1**3*ek*q(3)*q(5)+3*ek**2*q(2)*q(5)-(8*p1**2*ek*q(2)*q(5)
. )+7*p1**ek**2*q(4)**2-(14*p1**3*ek*q(4)**2)+7*p1**2*ek**2*q(1)*q(4)-(44*
. p1**4*ek*q(1)*q(4))-(4*p1**5*ek*q(3)**2)+8*p1**4*ek*q(2)*q(3)-(6*p1**5*q
. (2)**2)-(18*p1**7*q(1)**2))/3.0
T0=T0+p1**3*ek*q(6)**2*q(7)**2+10*ek*q(7)**2-(12*p1**2*q(7)
. **2)+7*ek*q(6)**2*q(5)*q(7)-(8*p1**2*q(6)**2*q(5)*q(7))+13*ek*q(5)*q(7)-
. (8*p1**2*q(5)*q(7))-(2*p1**2*ek*q(3)*q(7))-(5*p1*ek*q(2)*q(7))+16*p1**3
. *q(2)*q(7)+30*p1**2*ek*q(6)**2-(36*p1**4*q(6)**2)+44*p1**2*ek*q(4)*q(6
. )-(24*p1**4*q(4)*q(6))-(16*p1**3*ek*q(1)*q(6))+48*p1**5*q(1)*q(6)+3*ek
. *q(5)**2-(2*p1**2*ek*q(3)*q(5))-(5*p1*ek*q(2)*q(5))+8*p1**3*q(2)*q(5)+
. 14*p1**2*ek*q(4)**2-(16*p1**3*ek*q(1)*q(4))+24*p1**5*q(1)*q(4)-(4*p1**4
. *q(2)**2)-(12*p1**6*q(1)**2))/3.0
T0=T0+p1**2*fe*(6*p1*q(6)**2*q(7)**2+7*p1*q(7)**2+8*ek*q(6)*q(6)*q(7)-
. (4*p1**2*q(6)*q(6)*q(7))+4*ek*q(6)*q(4)*q(7)-(12*p1**2*q(6)*q(4)*q(7))-(4*
. p1*ek*q(6)**2*q(3)*q(7))+8*p1**3*q(6)**2*q(3)*q(7)-(8*p1*ek*q(3)*q(7))-(4*
. p1**3*q(3)*q(7))+4*ek*q(6)**2*q(2)*q(7)-(2*p1**2*q(6)**2*q(2)*q(7))+8*ek*
. q(2)*q(7)-(10*p1**2*q(2)*q(7))-(10*p1**3*q(6)*q(1)*q(7))+21*p1**3*q(6)
. **2+20*p1*ek*q(4)*q(6)+4*p1**3*q(4)*q(6)+20*p1**2*ek*q(1)*q(6)-(38*p1**
. 4*q(1)*q(6))-(4*p1*ek*q(3)*q(5))+4*ek*q(2)*q(5)+23*p1*ek*q(4)**2-(32*p1
. **3*q(4)**2)+26*p1**2*ek*q(1)*q(4)-(68*p1**4*q(1)*q(4))+5*p1**3*ek*q(3
. )**2-(8*p1**5*q(3)**2)-(6*p1**2*ek*q(2)*q(3))+20*p1**4*q(2)*q(3)+p1*ek*
. q(2)**2-(5*p1**3*q(2)**2)+3*p1**3*ek*q(1)**2-(15*p1**5*q(1)**2))/6.0
bmn2(1,6)=T0+p1**3*ek**2*el*(3*ek*q(7)**2-(16*p1**2*q(7)**2)-(12*p1
. **2*q(6)**2*q(5)*q(7))+6*ek*q(5)*q(7)-(20*p1**2*q(5)*q(7))+12*p1**3*q(
. 2)*q(7)+9*p1**2*ek*q(6)**2-(48*p1**4*q(6)**2)+18*p1**2*ek*q(4)*q(6)-(6
. 64*p1**4*q(4)*q(6))+32*p1**5*q(1)*q(6)+3*ek*q(5)**2-(4*p1**2*q(5)**2)
. +12*p1**3*q(2)*q(5)+9*p1**2*ek*q(4)**2-(16*p1**4*q(4)**2)+32*p1**5*q(1
. )*q(4))/3.0+p1**2*de*(12*q(6)*q(6)*q(7)+8*q(6)*q(4)*q(7)-(6*p1*q(6)
. **2*q(3)*q(7))-(6*p1*q(3)*q(7))+6*q(6)**2*q(2)*q(7)+6*q(2)*q(7)+2*p1*
. q(6)*q(1)*q(7)+14*p1*q(4)*q(6)+14*p1**2*q(1)*q(6)+17*p1*q(4)**2+20*p1**
. 2*q(1)*q(4)+7*p1**3*q(3)**2-(8*p1**2*q(2)*q(3))+p1*q(2)**2+3*p1**3*q(
. 1)**2)/6.0
T0=p1*he*(8*p1**2*ek*q(6)**2*q(6)*q(7)+20*p1**2*ek*q(6)*q(7)-(12*p1**4
. *q(6)*q(7))-(4*p1**2*ek*q(6)**2*q(4)*q(7))+ek**2*q(4)*q(7)+8*p1**2*ek*q(
. 4)*q(7)-(12*p1**3*ek*q(6)**2*q(1)*q(7))+p1*ek**2*q(1)*q(7)-(12*p1**3*ek*q
. (1)*q(7))+12*p1**5*q(1)*q(7)-(2*p1**3*ek*q(6)*q(6)**2)+6*p1**2*ek*q(6)**2*
. q(5)*q(6)+6*p1**2*ek*q(5)*q(6)-(2*p1**3*ek*q(6)*q(4)*q(6))+4*p1**4*ek*q(6
. )**2*q(3)*q(6)-(3*p1**2*ek**2*q(3)*q(6))+2*p1**4*ek*q(3)*q(6)-(2*p1**3*
. ek*q(6)**2*q(2)*q(6))+3*p1*ek**2*q(2)*q(6)-(16*p1**3*ek*q(2)*q(6))+12*p1
. **5*q(2)*q(6)+ek**2*q(4)*q(5)-(6*p1**3*ek*q(1)*q(5))+p1*ek**2*q(
. 1)*q(5)-(6*p1**3*ek*q(1)*q(5))+4*p1**4*ek*q(6)**2*q(3)*q(4)-(3*p1**2*ek**
. 2*q(3)*q(4))+8*p1**4*ek*q(3)*q(4)-(2*p1**3*ek*q(6)**2*q(2)*q(4))+3*p1*ek
. **2*q(2)*q(4)-(16*p1**3*ek*q(2)*q(4))+6*p1**5*ek*q(1)*q(3)-(12*p1**6*q
. (1)*q(2)))/3.0
T1=6*p1**2*q(6)**2*q(6)*q(7)+7*p1**2*q(6)*q(7)+2*ek*q(6)**2*q(4)*q(7)-(12
. *p1**2*q(6)**2*q(4)*q(7))+2*ek*q(4)*q(7)-(2*p1**2*q(4)*q(7))+2*p1*ek*q(6)
. **2*q(1)*q(7)-(18*p1**3*q(6)**2*q(1)*q(7))+2*p1*ek*q(1)*q(7)-(9*p1**3*q
. (1)*q(7))+2*p1*ek*q(6)*q(6)**2-(p1**3*q(6)*q(6)**2)+2*p1*ek*q(6)*q(4)*q(6)-
. 6*p1**3*q(6)*q(4)*q(6)-(2*p1**2*ek*q(6)**2*q(3)*q(6))+4*p1**4*q(6)**2*q(3)
. *q(6)-(4*p1**2*ek*q(3)*q(6))-(2*p1**4*q(3)*q(6))+2*p1*ek*q(6)**2*q(2)*q(
. 6)-(p1**3*q(6)**2*q(2)*q(6))+4*p1*ek*q(2)*q(6)-(5*p1**3*q(2)*q(6))-(5*p1
. **4*q(6)*q(1)*q(6))+ek*q(6)**2*q(4)*q(5)+ek*q(4)*q(5)+p1*ek*q(6)**2*q(1)*q(
. 5)+p1*ek*q(1)*q(5)-(2*p1**2*ek*q(6)**2*q(3)*q(4))-(6*p1**2*ek*q(3)*q(4))+
. 12*p1**4*q(3)*q(4)+2*p1*ek*q(6)**2*q(2)*q(4)-(6*p1**3*q(6)**2*q(2)*q(4))+
. 5*p1*ek*q(2)*q(4)-(10*p1**3*q(2)*q(4))-(4*p1**5*q(6)**2*q(1)*q(3))-(2*p1
. **3*ek*q(1)*q(3))+14*p1**5*q(1)*q(3)
T0=T0+p1**3*ek*ej*(6*ek*q(6)**2*q(6)*q(7)+20*ek*q(6)*q(7)-(24*p1**2*q
. (6)*q(7))+6*ek*q(6)**2*q(4)*q(7)+14*ek*q(4)*q(7)-(8*p1**2*q(4)*q(7))-
. 6*p1*ek*q(1)*q(7))+16*p1**3*q(1)*q(7)+7*ek*q(6)**2*q(5)*q(6)-(8*p1**2*q(6)
. **2*q(5)*q(6))+13*ek*q(5)*q(6)-(8*p1**2*q(5)*q(6))-(2*p1**2*ek*q(3)*q

```

```

. (6))-(5*p1**ek*q(2)*q(6))+16*p1**3*q(2)*q(6)+7*ek*q(6)**2*q(4)*q(5)+7*ek*
. q(4)*q(5)+8*p1**3*q(6)**2*q(1)*q(5)-(6*p1**ek*q(1)*q(5))+8*p1**3*q(1)*q(
. 5)-(2*p1**2*ek*q(3)*q(4))-(5*p1**ek*q(2)*q(4))+8*p1**3*q(2)*q(4)-(8*p1**4*q(1)*q(2)))/3.0+p1**fe*(T1-(5*p1**4*q(6)**2*q(1)*q(2))+p1**2*ek*q(
. 1)*q(2)-(5*p1**4*q(1)*q(2)))/3.0
bmn2(1,7)=T0+2.0/3.0*p1**3*q(6)**2*el*(3*q(6)*q(7)-(16*p1**2*q(6)
. *q(7))+3*q(6)*q(7)-(12*p1**2*q(4)*q(7))+4*p1**3*q(1)*q(7)-(6*p1**2
. *q(6)**2*q(5)*q(6))+3*q(6)*q(6)-(10*p1**2*q(5)*q(6))+6*p1**3*q(2)*q(
. 6)-(6*p1**2*q(6)**2*q(4)*q(5))+3*q(6)*q(5)-(6*p1**2*q(4)*q(5))+4*p1
. **3*q(1)*q(5)+6*p1**3*q(2)*q(4))+p1**de*(2*q(6)**2*q(4)*q(7)+q(4)
. *q(7)+2*p1*q(6)**2*q(1)*q(7)+p1*q(1)*q(7)+3*p1*q(6)*q(6)**2+4*p1*q(6)*q(4)*
. q(6)-(3*p1**2*q(6)**2*q(3)*q(6))-(3*p1**2*q(3)*q(6))+3*p1*q(6)**2*q(2)*q(
. 6)+3*p1*q(2)*q(6)+p1**2*q(6)*q(1)*q(6)-(3*p1**2*q(6)**2*q(3)*q(4))-(5*p1
. **2*q(3)*q(4))+4*p1*q(6)**2*q(2)*q(4)+4*p1*q(2)*q(4)-(2*p1**3*q(1)*q(3
. ))+p1**2*q(6)**2*q(1)*q(2)+p1**2*q(1)*q(2))/3.0-(8.0/3.0*p1**5*ek**3*re
. *(q(6)+q(4))*(q(7)+q(5)))
T0=4.0/3.0*p1**4*ek**2*el*(4*(q(7)+q(5))**2+p1**2*(q(6)+q(4))**2)
. +fe*(6*p1**2*q(6)**4*q(7)**2+10*ek*q(6)**2*q(7)**2-(22*p1**2*q(6)**2*q
. (7)**2)+10*ek*q(7)**2-(19*p1**2*q(7)**2)-(10*p1**3*q(6)*q(6)*q(7))+10*
. ek*q(6)**2*q(5)*q(7)+10*ek*q(5)*q(7)+16*p1**4*q(6)**2*q(3)*q(7)-(16*p1**2
. *ek*q(3)*q(7))+68*p1**4*q(3)*q(7)-(30*p1**3*q(6)**2*q(2)*q(7))+6*p1**ek*q(
. 2)*q(7)-(30*p1**3*q(2)*q(7))+2*p1**2*ek*q(6)**2-(9*p1**4*q(6)**2)+6*
. p1**2*ek*q(4)*q(6)-(28*p1**4*q(4)*q(6))+2*p1**3*ek*q(1)*q(6)-(10*p1**5*
. q(1)*q(6))-(16*p1**2*ek*q(3)*q(5))+6*p1**ek*q(2)*q(5)+4*p1**2*ek*q(4)**2
. +8*p1**4*q(4)**2+2*p1**3*ek*q(1)*q(4)+44*p1**5*q(1)*q(4)+32*p1**6*q(3)
. **2-(132*p1**5*q(2)*q(3))+81*p1**4*q(2)**2+27*p1**6*q(1)**2)/6.0
T0=T0+de*(6*q(6)**4*q(7)**2+14*q(6)**2*q(7)**2+7*q(7)**2+2*p1*q(6)*q(
. 6)*q(7)-(16*p1**2*q(6)**2*q(3)*q(7))-(20*p1**2*q(3)*q(7))+6*p1*q(6)**2*
. q(2)*q(7)+6*p1*q(2)*q(7)+p1**2*q(6)**2+4*p1**2*q(4)*q(6)+2*p1**3*q(1)
. *q(6)-(3*p1**2*q(4)**2)-(10*p1**3*q(1)*q(4))-(5*p1**4*q(3)**2)+30*p1
. **3*q(2)*q(3)-(18*p1**2*q(2)**2)-(6*p1**4*q(1)**2))/6.0
T0=T0-(1.0/6.0*he*(20*p1**2*ek*q(6)**2*q(7)**2-(7*ek**2*q(7)**2)+44
. *p1**2*ek*q(7)**2-(36*p1**4*q(7)**2)+20*p1**2*ek*q(6)**2*q(5)*q(7)-(14*ek
. **2*q(5)*q(7))+44*p1**2*ek*q(5)*q(7)-(44*p1**4*ek*q(3)*q(7))+72*p1**5*
. q(2)*q(7)-(p1**2*ek**2*q(6)**2)+12*p1**4*ek*q(6)**2-(12*p1**6*q(6)**2)
. -(2*p1**2*ek**2*q(4)*q(6))+24*p1**4*ek*q(4)*q(6)+24*p1**7*q(1)*q(6)-(7
. *ek**2*q(5)**2)-(44*p1**4*ek*q(3)*q(5))-(p1**2*ek**2*q(4)**2)+12*p1**4*
. ek*q(4)**2-(36*p1**6*q(2)**2)-(12*p1**8*q(1)**2)))
bmn2(2,2)=T0-(1.0/3.0*p1**2*ek*ej*(8*ek*q(7)**2-(24*p1**2*q(7)**2)
. -(8*p1**2*q(6)**2*q(5)*q(7))+16*ek*q(5)*q(7)-(24*p1**2*q(5)*q(7))+24*p1
. **3*q(2)*q(7)+3*p1**2*ek*q(6)**2-(8*p1**4*q(6)**2)+6*p1**2*ek*q(4)*q(6
. )-(8*p1**4*q(4)*q(6))+8*p1**5*q(1)*q(6)+8*ek*q(5)**2+24*p1**3*q(2)*q(
. 5)+3*p1**2*ek*q(4)**2+8*p1**5*q(1)*q(4))+ae*(q(4)**2+2*p1*q(1)*q
. (4)+3*p1**2*q(3)**2-(6*p1*q(2)*q(3))+3*q(2)**2+p1**2*q(1)**2)/2.0
T0=-(1.0/6.0*p1**de*(6*q(6)**4*q(7)**2+14*q(6)**2*q(7)**2+7*q(7)**2
. +8*p1*q(6)*q(7)-(26*p1**2*q(6)**2*q(3)*q(7))-(34*p1**2*q(3)*q(7))+
. 16*p1*q(6)**2*q(2)*q(7)+20*p1*q(2)*q(7)+p1**2*q(6)**2+10*p1**2*q(4)*q(6
. )+8*p1**3*q(1)*q(6)+4*p1**2*q(4)**2-(2*p1**3*q(1)*q(4))+12*p1**4*q(3)
. **2+10*p1**3*q(2)*q(3)-(15*p1**2*q(2)**2)-(5*p1**4*q(1)**2)))
T0=T0-(1.0/3.0*p1**fe*(5*ek*q(6)**2*q(7)**2-(6*p1**3*q(6)*q(7))+5*ek*q(6)**2*q(5)*q(7)
. +5*ek*q(5)*q(7)-(13*p1**2*ek*q(3)*q(7))+32*p1**4*q(3)*q(7)-(8*p1**3*
. q(6)**2*q(2)*q(7))+8*p1**ek*q(2)*q(7)-(34*p1**3*q(2)*q(7))+p1**2*ek*q(6)
. **2-(p1**4*q(6)**2)+5*p1**2*ek*q(4)*q(6)-(12*p1**4*q(4)*q(6))+3*p1**3*
. ek*q(1)*q(6)-(10*p1**5*q(1)*q(6))-(13*p1**2*ek*q(3)*q(5))+8*p1*ek*q(2)*
. q(5)+4*p1**2*ek*q(4)**2+3*p1**3*ek*q(1)*q(4)+12*p1**5*q(1)*q(4)-(32*p1
. **5*q(2)*q(3))+33*p1**4*q(2)**2+11*p1**6*q(1)**2))
bmn2(2,3)=T0+p1*ek*he*(8*p1**2*q(6)**2*q(7)**2-(7*ek*q(7)**2)+4*p1**2
. *q(7)**2-(14*ek*q(5)*q(7))+4*p1**2*q(5)*q(7)-(48*p1**4*q(3)*q(7))+44
. *p1**3*q(2)*q(7)-(p1**2*ek*q(6)**2)-(2*p1**2*ek*q(4)*q(6))+16*p1**4*q(4
. )*q(6)+16*p1**5*q(1)*q(6)-(7*ek*q(5)**2)-(48*p1**4*q(3)*q(5))+44*p1**3*
. 3*q(2)*q(5)-(p1**2*ek*q(4)**2)+16*p1**4*q(4)**2+16*p1**5*q(1)*q(4))/
. 6.0-(2.0/3.0*p1**3*ek**2*ej*(q(7)+q(5)**2)-(1.0/2.0*p1**ae*(q(4)**2+2*p1*q(1)*q(3))
. +3*p1**2*p1*q(1)*q(4)+3*p1**2*q(3)**2-(6*p1*q(2)*q(3))+3*q(2)**2+p1
. **2*q(1)**2))
T0=p1**3*ek*ej*(6*ek*q(6)**2*q(6)*q(7)+15*ek*q(6)*q(7)-(8*p1**2*q(6)*
. q(7))+6*ek*q(6)**2*q(4)*q(7)+10*ek*q(4)*q(7)-(5*p1*ek*q(1)*q(7))+8*p1**3
. *q(1)*q(7)+9*ek*q(5)*q(6)-(6*p1*ek*q(2)*q(6))+8*p1**3*q(2)*q(6)+4*ek*q
. (4)*q(5)-(5*p1*ek*q(1)*q(5))-(6*p1*ek*q(2)*q(4))-(8*p1**4*q(1)*q(2))/
. 3.0+p1**fe*(2*ek*q(6)**2*q(6)*q(7)+6*p1**2*q(6)**2*q(6)*q(7)+4*ek*q(6)
. *q(7)+2*p1**2*q(6)*q(7)+4*ek*q(6)**2*q(4)*q(7)+9*ek*q(4)*q(7)-(8*p1**2*
. q(4)*q(7))+2*p1*ek*q(6)**2*q(1)*q(7)-(6*p1**3*q(6)**2*q(1)*q(7))+5*p1*ek*q

```

```

. (1)*q(7)-(10*p1**3*q(1)*q(7))+2*ek*q(5)*q(6)-(5*p1**2*ek*q(3)*q(6))+
. 12*p1**4*q(3)*q(6)+3*p1*ek*q(2)*q(6)-(14*p1**3*q(2)*q(6))+5*ek*q(4)*q(
. 5)+3*p1*ek*q(1)*q(5)-(8*p1**2*ek*q(3)*q(4))+4*p1*ek*q(2)*q(4)+8*p1**3*q(
. 2)*q(4)-(3*p1**3*ek*q(1)*q(3))-(12*p1**5*q(1)*q(3))+p1**2*ek*q(1)*q(2)
. +22*p1**4*q(1)*q(2))/3.0
T0=T0+p1*ek*he*(10*p1**2*q(6)**2*q(6)*q(7)+3*ek*q(6)*q(7)+4*p1**2*q(6)
. )*q(7)+8*p1**2*q(6)**2*q(4)*q(7)+6*ek*q(4)*q(7)-(12*p1**2*q(4)*q(7))-
. 2*p1**3*q(6)**2*q(1)*q(7))+3*p1*ek*q(1)*q(7)-(16*p1**3*q(1)*q(7))+3*ek*q(
. 5)*q(6)-(p1**2*ek*q(3)*q(6))+8*p1**4*q(3)*q(6)+p1*ek*q(2)*q(6)-(12*p1
. **3*q(2)*q(6))+6*ek*q(4)*q(5)-(8*p1**2*q(4)*q(5))+3*p1*ek*q(1)*q(5)-
. 8*p1**3*q(1)*q(5))-(p1**2*ek*q(3)*q(4))+16*p1**4*q(3)*q(4)+p1*ek*q(2)*q(
. 4)-(12*p1**3*q(2)*q(4))+8*p1**5*q(1)*q(3))/3.0+p1*de*(3*q(6)**2*
. q(6)*q(7)+3*q(6)*q(7)+7*q(6)**2*q(4)*q(7)+7*q(4)*q(7)+4*p1*q(6)**2*q(1
. )*q(7)+4*p1*q(1)*q(7)-(5*p1**2*q(3)*q(6))+2*p1*q(2)*q(6)-(4*p1**2*q(3
. )*q(4))-(3*p1*q(2)*q(4))+p1**3*q(1)*q(3)-(5*p1**2*q(1)*q(2))/3.0
bmn2(2,4)=T0+2.0/3.0*p1**3*q(6)*q(7)-(10*p1**2*q(6)
. *q(7))+3*ek*q(4)*q(7)-(4*p1**2*q(4)*q(7))+6*p1**3*q(1)*q(7)+3*ek*q(5)
. *q(6)-(6*p1**2*q(5)*q(6))+4*p1**3*q(2)*q(6)+3*ek*q(4)*q(5)+6*p1**3*q(
. 1)*q(5)+4*p1**3*q(2)*q(4))-(8.0/3.0*p1**5*q(1)*q(3)*q(6)+q(4))*(q(
. 7)+q(5))-(ae*(p1*q(3)-q(2))*(q(4)+p1*q(1)))
T0=-(4.0/3.0*p1**3*q(6)*q(7)+q(5))**2+p1**2*(q(6)+q(4))**
. 2)+ek*he*(6*p1*q(6)**4*q(7)**2+14*p1*q(6)**2*q(7)**2+10*p1*q(7)**2-(2
. 2*p1**2*q(6)*q(6)*q(7))-(7*p1*ek*q(3)*q(7))+2*p1**3*q(3)*q(7)-(10*p1**2*
. q(6)**2*q(2)*q(7))+7*ek*q(2)*q(7)-(22*p1**2*q(2)*q(7))+4*p1**3*q(6)**2
. +3*p1*ek*q(4)*q(6)+3*p1**2*ek*q(1)*q(6)-(8*p1**4*q(1)*q(6))-(7*p1*ek*q(3
. )*q(5))+7*ek*q(2)*q(5)+3*p1*ek*q(4)**2-(4*p1**3*q(4)**2)+3*p1**2*ek*q(1
. )*q(4)-(8*p1**4*q(1)*q(4))-(12*p1**5*q(3)**2)+22*p1**4*q(2)*q(3))/3.0
T0=T0+p1*ek*ej*(16*ek*q(6)**2*q(7)**2-(8*p1**2*q(6)**2*q(7)**2)+20*ek*q(
. 7)**2-(12*p1**2*q(7)**2)+20*ek*q(6)**2*q(5)*q(7)+20*ek*q(5)*q(7)-(4*p1
. **2*ek*q(3)*q(7))+8*p1**3*q(6)**2*q(2)*q(7)-(16*p1*ek*q(2)*q(7))+24*p1**2
. 3*q(2)*q(7)+7*p1**2*ek*q(6)**2-(4*p1**4*q(6)**2)+9*p1**2*ek*q(4)*q(6)-
. (5*p1**3*q(1)*q(6))+8*p1**5*q(1)*q(6)-(4*p1**2*ek*q(3)*q(5))-(16*p1*
. ek*q(2)*q(5))+2*p1**2*ek*q(4)**2-(5*p1**3*ek*q(1)*q(4))-(12*p1**4*q(2)
. **2)-(4*p1**6*q(1)**2))/3.0
bmn2(2,5)=T0-(1.0/3.0*p1*ek**2*el*(12*p1**2*q(6)**2*q(7)**2-(9*ek*q(
. 7)**2)+32*p1**2*q(7)**2+24*p1**2*q(6)**2*q(5)*q(7)-(18*ek*q(5)*q(7))+
. 32*p1**2*q(5)*q(7)-(32*p1**3*q(2)*q(7))-(3*p1**2*ek*q(6)**2)+12*p1**4*
. q(6)**2-(6*p1**2*ek*q(4)*q(6))+12*p1**4*q(4)*q(6)-(12*p1**5*q(1)*q(6)
. )-(9*ek*q(5)**2)-(32*p1**3*q(2)*q(5))-(3*p1**2*ek*q(4)**2)-(12*p1**5*q
. (1)*q(4)))+ek*fe*(4*q(6)*q(6)*q(7)-(10*p1*q(6)**2*q(3)*q(7))-(10*p1
. *q(3)*q(7))+10*q(6)**2*q(2)*q(7)+10*q(2)*q(7)+4*p1*q(4)*q(6)+4*p1**2*
. q(1)*q(6)+5*p1*q(4)**2+6*p1**2*q(1)*q(4)+13*p1**3*q(3)**2-(16*p1**2*q
. (2)*q(3))+3*p1*q(2)**2+p1**3*q(1)**2)/6.0
T0=fe*(3*p1**2*q(6)**3*q(7)**2+2*ek*q(6)*q(7)**2-(p1**2*q(6)*q(7)**2)+7
. *p1**3*q(6)**2*q(6)*q(7)+7*p1**3*q(6)*q(7)+2*ek*q(6)*q(5)*q(7)+2*p1*ek*q(6)
. **2*q(4)*q(7)+6*p1**3*q(6)*q(7)-5*p1**2*q(4)*q(7)+4*p1*ek*q(4)*q(7)+2*p1**3*q(4)*
. q(7)+6*p1**4*q(6)*q(3)*q(7)-(5*p1**3*q(6)*q(2)*q(7))+2*p1**2*ek*q(6)**2*q(1
. )*q(7)-(p1**4*q(6)**2*q(1)*q(7))+4*p1**2*ek*q(1)*q(7)-(5*p1**4*q(1)*q(7)
. )-(2*p1**3*ek*q(3)*q(6))+2*p1**5*q(3)*q(6)+2*p1**2*ek*q(2)*q(6)-(9*p1
. **4*q(2)*q(6))+2*p1*ek*q(4)*q(5)+2*p1**2*ek*q(1)*q(5)-(5*p1**3*ek*q(3)*
. q(4))+12*p1**5*q(3)*q(4)+3*p1**2*ek*q(2)*q(4)-(14*p1**4*q(2)*q(4))-(3
. *p1**4*ek*q(1)*q(3))+10*p1**6*q(1)*q(3)+p1**3*ek*q(1)*q(2)-(5*p1**5*q(1
. )*q(2)))/3.0
T0=T0+de*(3*q(6)**3*q(7)**2+3*q(6)*q(7)**2+3*p1*q(6)**2*q(4)*q(7)+3*
. p1*q(4)*q(7)-(4*p1**2*q(6)*q(3)*q(7))+p1*q(6)*q(2)*q(7)+3*p1**2*q(6)**2*q(1
. )*q(7)+3*p1**2*q(1)*q(7)-(p1**3*q(3)*q(6))+p1**2*q(2)*q(6)-(5*p1**3*q
. (3)*q(4))+2*p1**2*q(2)*q(4)-(4*p1**4*q(1)*q(3))+p1**3*q(1)*q(2))/3.0
T0=T0-(1.0/3.0*p1*he*(2*p1*ek*q(6)*q(7)**2-(12*p1**2*ek*q(6)**2*q(6)*q(
. 7))-(20*p1**2*ek*q(6)*q(7))+12*p1**4*q(6)*q(7)+2*p1*ek*q(6)*q(5)*q(7)-(1
. 10*p1**2*ek*q(6)**2*q(4)*q(7))-(3*ek**2*q(4)*q(7))-(4*p1**2*ek*q(4)*q(7)
. )+2*p1**3*ek*q(6)**2*q(1)*q(7)-(3*p1*ek**2*q(1)*q(7))+16*p1**3*ek*q(1)*q(
. 7)-(12*p1**5*q(1)*q(7))-(8*p1**2*ek*q(5)*q(6))+p1**2*ek**2*q(3)*q(6)-(4
. *p1*ek**2*q(2)*q(6))+12*p1**3*ek*q(2)*q(6)-(12*p1**5*q(2)*q(6))-(3*ek**2
. *q(4)*q(5))-(3*p1*ek**2*q(1)*q(5))+8*p1**3*ek*q(1)*q(5)+p1**2*ek**2*q(3
. )*q(4)-(8*p1**4*ek*q(3)*q(4))-(p1*ek**2*q(2)*q(4))+12*p1**3*ek*q(2)*q(4
. )-(8*p1**5*ek*q(1)*q(3))+12*p1**6*q(1)*q(2)))
bmn2(2,6)=T0+p1**3*q(6)*q(7)+20*ek*q(6)*q(7)-(24
. *p1**2*q(6)*q(7))+6*ek*q(6)**2*q(4)*q(7)+15*ek*q(4)*q(7)-(8*p1**2*q(4)*
. q(7))-(5*p1*ek*q(1)*q(7))+16*p1**3*q(1)*q(7)+14*ek*q(5)*q(6)-(8*p1**2*
. q(5)*q(6))-(6*p1*ek*q(2)*q(6))+16*p1**3*q(2)*q(6)+9*ek*q(4)*q(5)-(5*p1
. *ek*q(1)*q(5))+8*p1**3*q(1)*q(5)-(6*p1*ek*q(2)*q(4))+8*p1**3*q(2)*q(4)

```

```

. -(8*p1**4*q(1)*q(2))/3.0+2.0/3.0*p1**3*ek**2*el*(3*ek*q(6)*q(7)-
. (16*p1**2*q(6)*q(7))+3*q(4)*q(7)-(10*p1**2*q(4)*q(7))+6*p1**3*q(1)
. *q(7)+3*ek*q(5)*q(6)-(12*p1**2*q(5)*q(6))+4*p1**3*q(2)*q(6)+3*ek*q(4)
. *q(5)-(6*p1**2*q(4)*q(5))+6*p1**3*q(1)*q(5)+4*p1**3*q(2)*q(4))-(8.0/
. 3.0*p1**5*ek**3*re*(q(6)+q(4))*(q(7)+q(5)))
T0=6*p1*ek*q(6)**4*q(7)**2+30*p1*ek*q(6)**2*q(7)**2+30*p1*ek*q(7)**2-(18*p1**2
. 3*q(7)**2)-(4*p1**2*ek*q(6)*q(7))+12*p1*ek*q(6)**4*q(5)*q(7)+28*p1*ek*
. q(6)**2*q(5)*q(7)+20*p1*ek*q(5)*q(7)+8*p1**3*ek*q(6)**2*q(3)*q(7)-(7*p1*ek
. **2*q(3)*q(7))+4*p1**3*ek*q(3)*q(7)-(20*p1**2*ek*q(6)**2*q(2)*q(7))+7*ek
. **2*q(2)*q(7)-(44*p1**2*ek*q(2)*q(7))+36*p1**4*q(2)*q(7)+6*p1**3*ek*q(6)
. **2*q(6)**2+10*p1**3*ek*q(6)**2-(6*p1**5*q(6)**2)-(2*p1**2*ek*q(6)*q(5)*
. q(6))+10*p1**3*ek*q(6)**2*q(4)*q(6)+3*p1*ek**2*q(4)*q(6)+4*p1**3*ek*q(4)*
. q(6)-(2*p1**4*ek*q(6)**2*q(1)*q(6))+3*p1**2*ek**2*q(1)*q(6)-(16*p1**4*ek*
. q(1)*q(6))+12*p1**6*q(1)*q(6)-(7*p1*ek**2*q(3)*q(5))+2*p1**3*ek*q(3)*q
. (5)-(10*p1**2*ek*q(6)**2*q(2)*q(5))+7*ek**2*q(2)*q(5)-(22*p1**2*ek*q(2)*
. q(5))+4*p1**3*ek*q(6)**2*q(4)**2+3*p1*ek**2*q(4)**2-(6*p1**3*ek*q(4)**2)
T1=18*p1*q(6)**4*q(7)**2+39*p1*q(6)**2*q(7)**2+21*p1*q(7)**2+12*p1**2*q(6)**2
. 3*q(6)*q(7)+8*ek*q(6)*q(7)-(4*p1**2*q(6)*q(7))-(20*p1*ek*q(6)**2*
. q(3)*q(7))+16*p1**3*q(6)**2*q(3)*q(7)-(20*p1*ek*q(3)*q(7))-(4*p1**3*q(3)
. )*q(7))+12*p1**2*q(6)**4*q(2)*q(7)+20*ek*q(6)**2*q(2)*q(7)-(44*p1**2*q(6)
. **2*q(2)*q(7))+20*ek*q(2)*q(7)-(38*p1**2*q(2)*q(7))+7*p1**3*q(6)**2*q(6)
. **2+7*p1**3*q(6)**2+4*ek*q(6)*q(5)*q(6)+4*p1*ek*q(6)**2*q(4)*q(6)+12*p1
. **3*q(6)**2*q(4)*q(6)+8*p1*ek*q(4)*q(6)+4*p1**3*q(4)*q(6)+12*p1**4*q(6)*q
. (3)*q(6)-(10*p1**3*q(6)*q(2)*q(6))+4*p1**2*ek*q(6)**2*q(1)*q(6)-(2*p1**4*
. q(6)**2*q(1)*q(6))+8*p1**2*ek*q(1)*q(6)-(10*p1**4*q(1)*q(6))-(10*p1*ek*
. q(6)**2*q(3)*q(5))-(10*p1*ek*q(3)*q(5))+10*ek*q(6)**2*q(2)*q(5)+10*ek*q(2)
. )*q(5)+4*p1**2*ek*q(6)**2*q(4)**2+9*p1*ek*q(4)**2-(8*p1**3*q(4)**2)+4*p1**2*
. ek*q(6)**2*q(1)*q(4)
T0=-(4.0/3.0*p1**3*ek**3*re*(3*(q(7)+q(5))**2+p1**2*(q(6)+q(4))**2
. 2))+he*(T0-(2*p1**4*ek*q(6)**2*q(1)*q(4))+3*p1**2*ek**2*q(1)*q(4)-(16*p1**4*ek*q(1)*q(4))-(12*p1**5*ek*q(3)**2)+22*p1**4*ek*q(2)*q(3)-(18*p1**5*q(2)**2)-(6*p1**7*q(1)**2))/3.0+fe*(T1-(12*p1**4*q(6)**2*q(1)
. )*q(4))+10*p1**2*ek*q(1)*q(4)-(20*p1**4*q(1)*q(4))+13*p1**3*ek*q(3)**2
. -(32*p1**5*q(3)**2)+16*p1**4*q(6)**2*q(2)*q(3)-(16*p1**2*ek*q(2)*q(3))+68*p1**4*q(2)*q(3)-(15*p1**3*q(6)**2*q(2)**2)+3*p1*ek*q(2)**2-(15*p1**3*q(2)**2)-(5*p1**5*q(6)**2*q(1)**2)+p1**3*ek*q(1)**2-(5*p1**5*q(1)**2))/6.0
T0=T0+p1*ek*ej*(18*ek*q(6)**2*q(7)**2+30*ek*q(7)**2-(36*p1**2*q(7)**2
. )+32*ek*q(6)**2*q(5)*q(7)-(16*p1**2*q(6)**2*q(5)*q(7))+40*ek*q(5)*q(7)-(24*p1**2*q(5)*q(7))-(4*p1**2*ek*q(3)*q(7))-(16*p1*ek*q(2)*q(7))+48*p1**3*q(2)*q(7)+3*p1**2*ek*q(6)**2*q(6)+10*p1**2*ek*q(6)**2*q(6)+12*p1**4*q(6)**2+6*p1**2*ek*q(6)**2*q(4)*q(6)-(5*p1**3*ek*q(1)*q(6))+16*p1**5*q(1)*q(6)+10*ek*q(6)**2*q(5)**2+10*ek*q(5)**2-(4*p1**2*ek*q(3)*q(5))+8*p1**3*q(6)**2*q(2)*q(5)-(16*p1*ek*q(2)*q(5))+24*p1**3*q(2)*q(5)+3*p1**2*ek*q(6)**2*q(4)**2+5*p1**2*ek*q(4)**2-2-(5*p1**3*ek*q(1)*q(4))+8*p1**5*q(1)*q(4)-(12*p1**4*q(2)**2)-(4*p1**6*q(1)**2))/3.0
T0=T0+p1*ek**2*el*(9*ek*q(7)**2-(48*p1**2*q(7)**2)-(24*p1**2*q(6)**2*
. q(5)*q(7))+18*ek*q(5)*q(7)-(64*p1**2*q(5)*q(7))+32*p1**3*q(2)*q(7)+3*p1**2*ek*q(6)**2-(16*p1**4*q(6)**2)+6*p1**2*ek*q(4)*q(6)-(20*p1**4*q(4)*q(6))+12*p1**5*q(1)*q(6)-(12*p1**2*q(6)**2*q(5)**2)+9*ek*q(5)**2-(16*p1**2*q(5)**2)+32*p1**3*q(2)*q(5)+3*p1**2*ek*q(4)**2-(4*p1**4*q(4)**2)+12*p1**5*q(1)*q(4))/3.0
bmnn2(2,7)=T0+de*(12*q(6)**3*q(6)*q(7)+12*q(6)*q(6)*q(7)-(12*p1*q(6)
. **4*q(3)*q(7))-(28*p1*q(6)**2*q(3)*q(7))-(14*p1*q(3)*q(7))+12*q(6)**4*q
. (2)*q(7)+28*q(6)**2*q(2)*q(7)+14*q(2)*q(7)+6*p1*q(6)**2*q(4)*q(6)+6*p1*
. q(4)*q(6)-(8*p1**2*q(6)*q(3)*q(6))+2*p1*q(6)*q(2)*q(6)+6*p1**2*q(6)**2*q(1)
. )*q(6)+6*p1**2*q(1)*q(6)+7*p1*q(6)**2*q(4)**2+7*p1*q(4)**2+8*p1**2*q(6)**2
. 2*q(1)*q(4)+8*p1**2*q(1)*q(4)+13*p1**3*q(6)**2*q(3)**2+17*p1**3*q(3)**2
. 2-(16*p1**2*q(6)**2*q(2)*q(3))-(20*p1**2*q(2)*q(3))+3*p1*q(6)**2*q(2)**2
. +3*p1*q(2)**2+p1**3*q(6)**2*q(1)**2+2*p1**3*q(1)**2)/6.0
T0=4.0/3.0*p1**4*ek**2*ej*(4*(q(7)+q(5))**2+p1**2*(q(6)+q(4))**2)
. +p1**2*de*(6*q(6)**4*q(7)**2+14*q(6)**2*q(7)**2+7*q(7)**2+14*p1*q(6)
. *q(6)*q(7)-(36*p1**2*q(6)**2*q(3)*q(7))-(48*p1**2*q(3)*q(7))+26*p1*q(6)
. **2*q(2)*q(7)+34*p1*q(2)*q(7)+p1**2*q(6)**2+16*p1**2*q(4)*q(6)+14*p1
. **3*q(1)*q(6)+12*p1**2*q(4)**2+8*p1**3*q(1)*q(4)+36*p1**4*q(3)**2-(24*p1**3*q(2)*q(3))-(5*p1**2*q(2)**2)-(3*p1**4*q(1)**2))/6.0
T0=T0+p1**2*fe*(5*ek*q(6)**2*q(7)**2+5*ek*q(7)**2+16*p1**2*q(7)**2+5
. *ek*q(6)**2*q(5)*q(7)+5*ek*q(5)*q(7)-(18*p1**2*ek*q(3)*q(7))+13*p1*ek*q(2)
. )*q(7)-(32*p1**3*q(2)*q(7))+p1**2*ek*q(6)**2+4*p1**4*q(6)**2+7*p1**2*ek
. *q(4)*q(6)+5*p1**3*ek*q(1)*q(6)-(8*p1**5*q(1)*q(6))-(18*p1**2*ek*q(3)*
. q(5))+13*p1*ek*q(2)*q(5)+6*p1**2*ek*q(4)**2+5*p1**3*ek*q(1)*q(4)+16*p1**2

```

```

. 4*q(2)**2+4*p1**6*q(1)**2)/3.0+p1**2*ek*he*(7*ek*q(7)**2+48*p1**2*
. q(7)**2+16*p1**2*q(6)**2*q(5)*q(7)+14*ek*q(5)*q(7)+48*p1**2*q(5)*q(7)-
. (48*p1**3*q(2)*q(7))+p1**2*ek*q(6)**2+16*p1**4*q(6)**2+2*p1**2*ek*q(4)*
. q(6)+16*p1**4*q(4)*q(6)-(16*p1**5*q(1)*q(6))+7*ek*q(5)**2-(48*p1**3*q
. (2)*q(5))+p1**2*ek*q(4)**2-(16*p1**5*q(1)*q(4))/6.0
bmn2(3,3)=T0+p1**2*a* (q(4)**2+2*p1*q(1)*q(4)+3*p1**2*q(3)**2-(6*
. p1*q(2)*q(3))+3*q(2)**2+p1**2*q(1)**2)/2.0
T0=-(1.0/3.0*p1**2*fe*(2*ek*q(6)**2*q(6)*q(7)+4*ek*q(6)*q(7)+12*p1**
. 2*q(6)*q(7)+4*ek*q(6)**2*q(4)*q(7)+10*ek*q(4)*q(7)+2*p1*ek*q(6)**2*q(1)*q
. (7)+6*p1*ek*q(1)*q(7)-(12*p1**3*q(1)*q(7))+2*ek*q(5)*q(6)-(7*p1**2*ek*q
. (3)*q(6))+5*p1*ek*q(2)*q(6)-(12*p1**3*q(2)*q(6))+6*ek*q(4)*q(5)+4*p1*ek
. *q(1)*q(5)-(12*p1**2*ek*q(3)*q(4))+8*p1*ek*q(2)*q(4)-(5*p1**3*ek*q(1)*q
. (3))+3*p1**2*ek*q(1)*q(2)+12*p1**4*q(1)*q(2)))
T0=T0-(1.0/3.0*p1**2*ek*he*(4*p1**2*q(6)**2*q(6)*q(7)+3*ek*q(6)*q(7)
. +24*p1**2*q(6)*q(7)+6*ek*q(4)*q(7)+16*p1**2*q(4)*q(7)-(4*p1**3*q(6)**2*
. q(1)*q(7))+3*p1*ek*q(1)*q(7)-(8*p1**3*q(1)*q(7))+3*ek*q(5)*q(6)+8*p1**
. 2*q(5)*q(6)-(p1**2*ek*q(3)*q(6))-(8*p1**4*q(3)*q(6))+p1*ek*q(2)*q(6)-
. 8*p1**3*q(2)*q(6))+6*ek*q(4)*q(5)+3*p1*ek*q(1)*q(5)-(8*p1**3*q(1)*q(5)
. )-(p1**2*ek*q(3)*q(4))+p1*ek*q(2)*q(4)-(16*p1**3*q(2)*q(4))+8*p1**5*q(1
. )*q(3)-(8*p1**4*q(1)*q(2)))-(1.0/3.0*p1**2*de*(3*q(6)**2*q(6)*q(7)
. (7)+3*q(6)*q(7)+6*q(6)**2*q(4)*q(7)+8*q(4)*q(7)+3*p1*q(6)**2*q(1)*q(7)
. +5*p1*q(1)*q(7)-(8*p1**2*q(3)*q(6))+5*p1*q(2)*q(6)-(12*p1**2*q(3)*q(4)
. )+4*p1*q(2)*q(4)-(4*p1**3*q(1)*q(3))-(p1**2*q(1)*q(2)))
bmn2(3,4)=T0-(2.0/3.0*p1**4*ek**2*ej*(2*q(6)**2*q(6)*q(7)+15*q(6)*
. q(7)+2*q(6)**2*q(4)*q(7)+16*q(4)*q(7)+p1*q(1)*q(7)+11*q(5)*q(6)-(4*p1
. **2*q(3)*q(6))+12*q(4)*q(5)+p1*q(1)*q(5)-(4*p1**2*q(3)*q(4)))-(8.0
. /3.0*p1**4*ek**3*el*(q(6)+q(4))*(q(7)+q(5)))+p1*ae*(p1*q(3)-q
. (2))*(q(4)+p1*q(1))
T0=-(4.0/3.0*p1**2*ek**3*el*(3*(q(7)+q(5))**2+p1**2*(q(6)+q(4))**
. 2))-(1.0/3.0*p1*ek*he*(4*p1*q(6)**4*q(7)**2+16*p1*q(6)**2*q(7)**2+13*
. p1*q(7)**2+4*p1**2*q(6)*q(7)-(8*p1**3*q(6)**2*q(3)*q(7))-(7*p1*ek*q(3
. )*q(7))-(24*p1**3*q(3)*q(7))+7*ek*q(2)*q(7)-(2*p1**2*q(2)*q(7))+5*p1
. **3*q(6)**2+3*p1**2*ek*q(4)*q(6)+8*p1**3*q(4)*q(6)+3*p1**2*ek*q(1)*q(6)-
. 2*p1**4*q(1)*q(6)-(7*p1*ek*q(3)*q(5))+7*ek*q(2)*q(5)+3*p1*ek*q(4)**2+3
. *p1**2*ek*q(1)*q(4)-(8*p1**4*q(1)*q(4))+24*p1**4*q(2)*q(3)-(11*p1**3*q
. (2)**2)-(3*p1**5*q(1)**2)))-(2.0/3.0*p1**2*ek**2*ej*(10*q(6)**2*q(4
. 7)**2+14*q(7)**2+12*q(6)**2*q(5)*q(7)+14*q(5)*q(7)-(16*p1**2*q(3)*q(6)
. )+2*p1*q(2)*q(7)+5*p1**2*q(6)**2+11*p1**2*q(4)*q(6)+p1**3*q(1)*q(6)
. -(16*p1**2*q(3)*q(5))+2*p1*q(2)*q(5)+6*p1**2*q(4)**2+2*p1**3*q(1)*q(4))
)
bmn2(3,5)=T0-(1.0/3.0*p1*ek*fe*(2*q(6)*q(6)*q(7)-(5*p1*q(6)**2*q(3)*q
. (7))-(5*p1*q(3)*q(7))+5*q(6)**2*q(2)*q(7)+5*q(2)*q(7)+2*p1*q(4)*q(6)+
. 2*p1**2*q(1)*q(6)+3*p1*q(4)**2+4*p1**2*q(1)*q(4)+9*p1**3*q(3)**2-(13*
. p1**2*q(2)*q(3))+4*p1*q(2)**2+p1**3*q(1)*q(2)))
T0=-(1.0/3.0*p1*de*(3*q(6)**3*q(7)**2+3*q(6)*q(7)**2+3*p1*q(6)**2*q(4
. )*q(7)+3*p1*q(4)*q(7)-(7*p1**2*q(6)*q(3)*q(7))+4*p1*q(6)*q(2)*q(7)+3*p1**
. 2*q(6)**2*q(1)*q(7)+3*p1**2*q(1)*q(7)-(p1**3*q(3)*q(6))+p1**2*q(2)*q(6
. )-(8*p1**3*q(3)*q(4))+5*p1**2*q(2)*q(4)-(7*p1**4*q(1)*q(3))+4*p1**3*q
. (1)*q(2)))
T0=T0-(1.0/3.0*p1*fe*(2*ek*q(6)*q(7)**2+6*p1**2*q(6)*q(7)**2+4*p1**3*
. q(6)**2*q(6)*q(7)+10*p1**3*q(6)*q(7)+2*ek*q(6)*q(5)*q(7)+2*p1*ek*q(6)**2*q(4
. )*q(7)+4*p1*ek*q(4)*q(7)+12*p1**3*q(4)*q(7)-(6*p1**3*q(6)*q(2)*q(7))+2
. *p1**2*ek*q(6)**2*q(1)*q(7)-(4*p1**4*q(6)**2*q(1)*q(7))+4*p1**2*ek*q(1)*q(6
. 7)+2*p1**4*q(1)*q(7)-(2*p1**3*ek*q(3)*q(6))-(8*p1**5*q(3)*q(6))+2*p1**
. 2*ek*q(2)*q(6)-(2*p1**4*q(2)*q(6))+2*p1*ek*q(4)*q(5)+2*p1**2*ek*q(1)*q(5
. 5)-(7*p1**3*ek*q(3)*q(4))+5*p1**2*ek*q(2)*q(4)-(12*p1**4*q(2)*q(4))-(5
. *p1**4*ek*q(1)*q(3))+8*p1**6*q(1)*q(3)+3*p1**3*ek*q(1)*q(2)-(10*p1**5*q
. (1)*q(2)))
bmn2(3,6)=T0-(1.0/3.0*p1**2*ek*he*(4*p1*q(6)*q(7)**2+8*p1**2*q(6)**2*q
. (6)*q(7)+26*p1**2*q(6)*q(7)+4*p1*q(6)*q(5)*q(7)+4*p1**2*q(6)**2*q(4)*q(7
. )+3*ek*q(4)*q(7)+24*p1**2*q(4)*q(7)-(4*p1**3*q(6)**2*q(1)*q(7))+3*p1*ek*
. q(1)*q(7)-(2*p1**3*q(1)*q(7))+10*p1**2*q(5)*q(6)-(p1**2*ek*q(3)*q(6))
. -(16*p1**4*q(3)*q(6))+p1*ek*q(2)*q(6)+3*ek*q(4)*q(5)+8*p1**2*q(4)*q(5)
. +3*p1*ek*q(1)*q(5)-(2*p1**3*q(1)*q(5))-(p1**2*ek*q(3)*q(4))-(8*p1**4*q(3
. 3)*q(4))+p1*ek*q(2)*q(4)-(8*p1**3*q(2)*q(4))+8*p1**5*q(1)*q(3)-(8*p1**
. 4*q(1)*q(2)))-(2.0/3.0*p1**4*ek**2*ej*(2*q(6)**2*q(6)*q(7)+14*q(7)
. 6)*q(7)+2*q(6)**2*q(4)*q(7)+15*q(4)*q(7)+p1*q(1)*q(7)+10*q(5)*q(6)-
. 4*p1**2*q(3)*q(6))+11*q(4)*q(5)+p1*q(1)*q(5)-(4*p1**2*q(3)*q(4)))-(8.0/3.0*p1
. **4*ek**3*el*(q(6)+q(4))*(q(7)+q(5)))
T1=-(4.0/3.0*p1**2*ek**3*el*(3*(q(7)+q(5))**2+p1**2*(q(6)+q(4))**
. 2))-(1.0/3.0*p1*ek*he*(36*p1*q(6)**2*q(7)**2+39*p1*q(7)**2+8*p1**2*
. q(6)*q(7)+8*p1*q(6)**4*q(5)*q(7)+32*p1*q(6)**2*q(5)*q(7)+26*p1*q(5)*q

```

```

. (7)-(7*p1*ek*q(3)*q(7))-(48*p1**3*q(3)*q(7))-(8*p1**2*q(6)**2*q(2)*q(7)
. )+7*ek*q(2)*q(7)-(4*p1**2*q(2)*q(7))+4*p1**3*q(6)**2*q(6)**2+13*p1**3*q
. (6)**2+4*p1**2*q(6)*q(5)*q(6)+4*p1**3*q(6)**2*q(4)*q(6)+3*p1*ek*q(4)*q(6)
. +24*p1**3*q(4)*q(6)-(4*p1**4*q(6)**2*q(1)*q(6))+3*p1**2*ek*q(1)*q(6)-(2
. *p1**4*q(1)*q(6))-(8*p1**3*q(6)**2*q(3)*q(5))-(7*p1*ek*q(3)*q(5))-(24*p1
. **3*q(3)*q(5))+7*ek*q(2)*q(5)-(2*p1**2*q(2)*q(5))+3*p1*ek*q(4)**2+8*p1
. **3*q(4)**2-(4*p1**4*q(6)**2*q(1)*q(4))+3*p1**2*ek*q(1)*q(4)-(8*p1**4*q
. (1)*q(4))+24*p1**4*q(2)*q(3)-(11*p1**3*q(2)**2)-(3*p1**5*q(1)**2))
T0=24*p1*q(6)**2*q(7)**2+15*p1*q(7)**2+4*ek*q(6)*q(6)*q(7)+12*p1**2*q(6)*q(6
. )*q(7)-(10*p1*ek*q(6)**2*q(3)*q(7))-(10*p1*ek*q(3)*q(7))-(32*p1**3*q(3)*
. q(7))+10*ek*q(6)**2*q(2)*q(7)-(8*p1**2*q(6)**2*q(2)*q(7))+10*ek*q(2)*q(7
. )+2*p1**2*q(2)*q(7)+2*p1**3*q(6)**2*q(6)**2+5*p1**3*q(6)**2+2*ek*q(6)*q(5
. )*q(6)+2*p1*ek*q(6)**2*q(4)*q(6)+4*p1*ek*q(4)*q(6)+12*p1**3*q(4)*q(6)-(6
. *p1**3*q(6)*q(2)*q(6))+2*p1**2*ek*q(6)**2*q(1)*q(6)-(4*p1**4*q(6)**2*q(1)*q
. (6))+4*p1**2*ek*q(1)*q(6)+2*p1**3*q(1)*q(6)-(5*p1*ek*q(6)**2*q(3)*q(5))-
. (5*p1*ek*q(3)*q(5))+5*ek*q(6)**2*q(2)*q(5)+5*ek*q(2)*q(5)+2*p1*ek*q(6)**2*q
. (4)**2+5*p1*ek*q(4)**2+2*p1**2*ek*q(6)**2*q(1)*q(4)+6*p1**2*ek*q(1)*q(4)-
. (12*p1**4*q(1)*q(4))+9*p1**3*q(3)*q(2)**2-(13*p1**2*ek*q(2)*q(3))+32*p1**4
. *q(2)*q(3)-(4*p1**3*q(6)**2*q(2)**2)+4*p1*ek*q(2)**2-(17*p1**3*q(2)**2
. )+2*p1**5*q(6)**2*q(1)**2
T1=T1-(1.0/3.0*p1*fe*(T0+p1**3*ek*q(1)**2-(7*p1**5*q(1)**2)))-(2.0
. /3.0*p1**2*ek**2*ej*(12*q(6)**2*q(7)**2+21*q(7)**2+20*q(6)**2*q(5)*
. q(7)+28*q(5)*q(7)-(16*p1**2*q(3)*q(7))+2*p1*q(2)*q(7)+p1**2*q(6)**2*q(
. 6)**2+7*p1**2*q(6)**2+2*p1**2*q(6)**2*q(4)*q(6)+15*p1**2*q(4)*q(6)+p1**3
. *q(1)*q(6)+6*q(6)**2*q(5)**2+7*q(5)**2-(16*p1**2*q(3)*q(5))+2*p1*q(2)
. )*q(5)+p1**2*q(6)**2*q(4)**2+8*p1**2*q(4)**2+2*p1**3*q(1)*q(4)))
bmn2(3,7)=T1-(1.0/3.0*p1*de*(6*q(6)**3*q(6)*q(7)+6*q(6)*q(6)*q(7)-
. (6*p1*q(6)**4*q(3)*q(7))-(14*p1*q(6)**2*q(3)*q(7))-(7*p1*q(3)*q(7))+6*q(6)
. **4*q(2)*q(7)+14*q(6)**2*q(2)*q(7)+7*q(2)*q(7)+3*p1*q(6)**2*q(4)*q(6)+
. 3*p1*q(4)*q(6)-(7*p1**2*q(6)*q(3)*q(6))+4*p1*q(6)*q(2)*q(6)+3*p1**2*q(6)**2
. *q(1)*q(6)+3*p1**2*q(1)*q(6)+3*p1*q(6)**2*q(4)**2+4*p1*q(4)**2+3*p1**2*
. q(6)**2*q(1)*q(4)+5*p1**2*q(1)*q(4)+9*p1**3*q(6)**2*q(3)**2+12*p1**3*q(3)
. )**2-(13*p1**2*q(6)**2*q(2)*q(3))-(17*p1**2*q(2)*q(3))+4*p1*q(6)**2*q(2)
. **2+5*p1*q(2)**2+p1**3*q(1)**2)
T0=2*p1**4*ek**4*te*((q(7)+q(6))**2+3*p1**2*(q(6)+q(4)**2)+2.0/
. 3.0*p1**2*ek**2*el*(ek*q(6)**2*q(7)**2+5*ek*q(7)**2+2*p1**2*q(7)**2+
. ek*q(6)**2*q(5)*q(7)+9*ek*q(5)*q(7)-(4*p1**2*ek*q(3)*q(7))+3*p1*ek*q(2)*q
. (7)-(4*p1**3*q(2)*q(7))+15*p1**2*ek*q(6)**2+8*p1**4*q(6)**2+39*p1**2*ek
. *q(4)*q(6)+9*p1**3*ek*q(1)*q(6)-(16*p1**5*q(1)*q(6))+4*ek*q(5)**2-(4*
. p1**2*ek*q(3)*q(5))+3*p1*ek*q(2)*q(5)+24*p1**2*ek*q(4)**2+9*p1**3*ek*q(1)
. *q(4)+2*p1**4*q(2)**2+8*p1**6*q(1)**2)
T0=T0+fe*(ek*q(6)**2*q(7)**2+2*ek*q(7)**2+4*p1**2*q(7)**2+4*p1*ek*q(6)*q(
. 6)*q(7)+ek*q(6)**2*q(5)*q(7)+ek*q(5)*q(7)-(4*p1**2*ek*q(6)**2*q(3)*q(7))-
. (10*p1**2*ek*q(3)*q(7))+4*p1*ek*q(6)**2*q(2)*q(7)+9*p1*ek*q(2)*q(7)-(8*p1
. **3*q(2)*q(7))+5*p1**2*ek*q(6)**2+16*p1**4*q(6)**2+33*p1**2*ek*q(4)*q(
. 6)+23*p1**3*ek*q(1)*q(6)-(32*p1**5*q(1)*q(6))-(6*p1**2*ek*q(3)*q(5))+5
. *p1*ek*q(2)*q(5)+36*p1**2*ek*q(4)**2+39*p1**3*ek*q(1)*q(4)+6*p1**4*ek*q(3)
. )**2-(8*p1**3*ek*q(2)*q(3))+2*p1**2*ek*q(2)**2+4*p1**4*q(2)**2+8*p1**4*
. ek*q(1)**2+16*p1**6*q(1)**2)/3.0
T0=T0+ek*he*(16*p1**2*q(6)**2*q(7)**2+ek*q(7)**2+36*p1**2*q(7)**2+16
. *p1**3*q(6)*q(7)+16*p1**2*q(6)**2*q(5)*q(7)+2*ek*q(5)*q(7)+16*p1**2*
. q(5)*q(7)-(12*p1**2*ek*q(3)*q(7))-(32*p1**4*q(3)*q(7))+16*p1**3*q(6)**2
. *q(2)*q(7)+12*p1*ek*q(2)*q(7)-(24*p1**3*q(2)*q(7))+7*p1**2*ek*q(6)**2+
. 100*p1**4*q(6)**2+42*p1**2*ek*q(4)*q(6)+144*p1**4*q(4)*q(6)+28*p1**3*ek
. *q(1)*q(6)-(56*p1**5*q(1)*q(6))+ek*q(5)**2-(12*p1**2*ek*q(3)*q(5))+12
. *p1*ek*q(2)*q(5)-(16*p1**3*q(2)*q(5))+42*p1**2*ek*q(4)**2+42*p1**3*ek*q(
. 1)*q(4)-(14*p1**5*q(1)*q(4))+p1**4*ek*q(3)**2-(2*p1**3*ek*q(2)*q(3))+
. 32*p1**5*q(2)*q(3)+p1**2*ek*q(2)**2-(12*p1**4*q(2)**2)+7*p1**4*ek*q(1)
. **2-(44*p1**6*q(1)**2))/6.0
T0=T0+p1**2*ek**2*ej*(14*q(6)**2*q(7)**2+25*q(7)**2+6*p1*q(6)*q(6)*q(
. 7)+20*q(6)**2*q(5)*q(7)+28*q(5)*q(7)-(4*p1**2*q(6)**2*q(3)*q(7))-(32*p1
. **2*q(3)*q(7))+6*p1*q(6)**2*q(2)*q(7)+10*p1*q(2)*q(7)+76*p1**2*q(6)**2
. +180*p1**2*q(4)*q(6)+28*p1**3*q(1)*q(6)+4*q(5)**2-(24*p1**2*q(3)*q(5))
. )+4*p1*q(2)*q(5)+96*p1**2*q(4)**2+12*p1**3*q(1)*q(4)+4*p1**4*q(3)**2
. -(3*p1**2*q(2)**2)-(8*p1**4*q(1)**2))/3.0+de*(2*q(6)**2*q(7)**2+
. q(7)**2+14*p1*q(6)*q(7)-(12*p1**2*q(6)**2*q(3)*q(7))-(16*p1**2*q(3)
. *q(7))+14*p1*q(6)**2*q(2)*q(7)+14*p1*q(2)*q(7)+7*p1**2*q(6)**2+48*p1**2
. *q(4)*q(6)+34*p1**3*q(1)*q(6)+36*p1**2*q(4)**2+24*p1**3*q(1)*q(4)+12
. *p1**4*q(3)**2-(8*p1**3*q(2)*q(3))-(3*p1**2*q(2)**2)-(5*p1**4*q(1)**2
. )/6.0
bmn2(4,4)=T0+p1**2*ek**3*re*(3*ek*q(7)**2+16*p1**2*q(7)**2+16*p1**2
. *q(6)**2*q(5)*q(7)+6*ek*q(5)*q(7)+16*p1**2*q(5)*q(7)-(16*p1**3*q(2)*q(

```

```

. 7))+9*p1**2*ek*q(6)**2+48*p1**4*q(6)**2+18*p1**2*ek*q(4)*q(6)+48*p1**4*
. q(4)*q(6)-(48*p1**5*q(1)*q(6))+3*ek*q(5)**2-(16*p1**3*q(2)*q(5))+9*p1
. **2*ek*q(4)**2-(48*p1**5*q(1)*q(4)))/6.0+aet*(3*q(4)**2+6*p1*q(1)
. *q(4)+p1**2*q(3)**2-(2*p1*q(2)*q(3))+q(2)**2+3*p1**2*q(1)**2)/2.0
T0=2.0/3.0*p1**2*ek**2*el*(ek*q(6)**2*q(6)*q(7)+6*p1**2*q(6)**2*q(6)*q
. (7)+6*ek*q(6)*q(7)+6*p1**2*q(6)*q(7)+ek*q(6)**2*q(4)*q(7)+9*ek*q(4)*q(7
. )-(6*p1**3*q(6)**2*q(1)*q(7))+3*p1*ek*q(1)*q(7)-(6*p1**3*q(1)*q(7))+5*ek
. *q(5)*q(6)-(4*p1**2*ek*q(3)*q(6))+3*p1*ek*q(2)*q(6)-(6*p1**3*q(2)*q(6)
. )+8*ek*q(4)*q(5)+3*p1*ek*q(1)*q(5)-(4*p1**2*ek*q(3)*q(4))+3*p1*ek*q(2)*q
. (4)+6*p1**4*q(1)*q(2))+p1**2*ek**3*re*(8*p1**2*q(6)**2*q(6)*q(7)+3*
. ek*q(6)*q(7)+16*p1**2*q(6)*q(7)+8*p1**2*q(6)**2*q(4)*q(7)+3*ek*q(4)*q(7
. )+8*p1**2*q(4)*q(7)-(8*p1**3*q(1)*q(7))+3*ek*q(5)*q(6)+8*p1**2*q(5)*q
. (6)-(8*p1**3*q(2)*q(6))+3*ek*q(4)*q(5)-(8*p1**3*q(1)*q(5))-(3*p1**3*q
. (2)*q(4)))/3.0
T0=T0+ek*he*(8*p1**2*q(6)**2*q(6)*q(7)+8*p1**2*q(6)*q(7)+8*p1**2*q(6)
. **2*q(4)*q(7)+ek*q(4)*q(7)+8*p1**2*q(4)*q(7)+p1*ek*q(1)*q(7)-(3*p1**2*
. ek*q(3)*q(6))-(8*p1**4*q(3)*q(6))+3*p1*ek*q(2)*q(6)+ek*q(4)*q(5)+p1*ek*q
. (1)*q(5)-(6*p1**2*ek*q(3)*q(4))+6*p1*ek*q(2)*q(4)-(8*p1**3*q(2)*q(4))-
. (3*p1**3*ek*q(1)*q(3))+8*p1**5*q(1)*q(3)+3*p1**2*ek*q(1)*q(2)-(8*p1**4*
. q(1)*q(2))/3.0+p1**2*ek**2*ej*(13*q(6)**2*q(6)*q(7)+21*q(6)*q(7)
. +20*q(6)**2*q(4)*q(7)+28*q(4)*q(7)+7*p1*q(6)**2*q(1)*q(7)+7*p1*q(1)*q(7
. )+8*p1*q(5)*q(6)-(22*p1**2*q(3)*q(6))+9*p1*q(2)*q(6)+8*p1*q(4)*q(5)-(24*p1
. **2*q(3)*q(4))+4*p1*q(2)*q(4)-(2*p1**3*q(1)*q(3))-(5*p1**2*q(1)*q(2)
. ))/3.0
bmn2(4,5)=T0+ek*fe*(q(6)**2*q(4)*q(7)+q(4)*q(7)+p1*q(6)**2*q(1)*q(7)
. +p1*q(1)*q(7)-(2*p1**2*q(3)*q(6))+2*p1*q(2)*q(6)-(6*p1**2*q(3)*q(4))+
. 5*p1*q(2)*q(4)-(4*p1**3*q(1)*q(3))+3*p1**2*q(1)*q(2))/3.0+4*p1**4*ek**2*
. 4*te*(q(6)+q(4))*(q(7)+q(5))
T0=2*p1**4*ek**4*te*((q(7)+q(5))**2+3*p1**2*(q(6)+q(4))**2)+p1**2*
. ek*ej*(11*ek*q(6)**2*q(7)**2+18*ek*q(7)**2+4*p1**2*q(7)**2+12*p1*ek*
. q(6)*q(6)*q(7)+13*ek*q(6)**2*q(5)*q(7)+21*ek*q(5)*q(7)+6*p1*ek*q(6)*q(4)*q(7
. )-(4*p1**2*ek*q(6)**2*q(3)*q(7))-(30*p1**2*ek*q(3)*q(7))+6*p1*ek*q(6)**2*q
. (2)*q(7)+15*p1*ek*q(2)*q(7)-(8*p1**3*q(2)*q(7))+54*p1**2*ek*q(6)**2+12
. *p1**4*q(6)**2+152*p1**2*ek*q(4)*q(6)+44*p1**3*ek*q(1)*q(6)-(24*p1**5*q
. (1)*q(6))+4*ek*q(5)**2-(22*p1**2*ek*q(3)*q(5))+9*p1*ek*q(2)*q(5)+90*p1
. **2*ek*q(4)**2+28*p1**3*ek*q(1)*q(4)+4*p1**4*ek*q(3)**2-(3*p1**2*ek*q(2)
. **2)+4*p1**4*q(2)**2-(8*p1**4*ek*q(1)**2)+12*p1**6*q(1)**2)/3.0
T0=T0+p1**2*ek**2*el*(2*ek*q(6)**2*q(7)**2+7*ek*q(7)**2+16*p1**2*q(7)
. **2+2*ek*q(6)**2*q(5)*q(7)+12*p1**2*q(6)**2*q(5)*q(7)+12*ek*q(5)*q(7)+12
. *p1**2*q(5)*q(7)-(8*p1**2*ek*q(3)*q(7))+6*p1*ek*q(2)*q(7)-(20*p1**3*q(2
. )*q(7))+21*p1**2*ek*q(6)**2+48*p1**4*q(6)**2+60*p1**2*ek*q(4)*q(6)+32*
. p1**4*q(4)*q(6)+18*p1**3*ek*q(1)*q(6)-(64*p1**5*q(1)*q(6))+5*ek*q(5)**2
. -2-(8*p1**2*ek*q(3)*q(5))+6*p1*ek*q(2)*q(5)-(12*p1**3*q(2)*q(5))+39*p1**2
. *ek*q(4)**2+18*p1**3*ek*q(1)*q(4)-(32*p1**5*q(1)*q(4))+4*p1**4*q(2)**2
. +216*p1**6*q(1)**2)/3.0
T0=T0+p1*ek*he*(20*p1*q(6)**2*q(7)**2+28*p1*q(7)**2+40*p1**2*q(6)*q(6)*
. q(7)+16*p1*q(6)**2*q(5)*q(7)+16*p1*q(5)*q(7)+16*p1**2*q(6)*q(4)*q(7)-(8*
. p1**3*q(6)**2*q(3)*q(7))-(6*p1*ek*q(3)*q(7))-(48*p1**3*q(3)*q(7))+20*p1
. **2*q(6)**2*q(2)*q(7)+6*ek*q(2)*q(7)+8*p1**2*q(2)*q(7)-(4*p1**3*q(6)*q(1
. )*q(7))+84*p1**3*q(6)**2+14*p1*ek*q(4)*q(6)+200*p1**3*q(4)*q(6)+14*p1
. **2*ek*q(1)*q(6)+32*p1**4*q(1)*q(6)-(6*p1*ek*q(3)*q(5))-(16*p1**3*q(3)
. *q(5))+6*ek*q(2)*q(5)+21*p1*ek*q(4)**2+72*p1**3*q(4)**2+28*p1**2*ek*q(1
. )*q(4)-(56*p1**4*q(1)*q(4))+p1**3*ek*q(3)**2+8*p1**5*q(3)**2-(2*p1**2*
. ek*q(2)*q(3))+16*p1**4*q(2)*q(3)+p1*ek*q(2)**2-(12*p1**3*q(2)**2)+7*p1
. **3*ek*q(1)**2-(44*p1**5*q(1)**2))/6.0
T0=T0+p1*fe*(12*p1*q(6)**2*q(7)**2+10*p1*q(7)**2+8*ek*q(6)*q(6)*q(7)+
. 24*p1**2*q(6)*q(6)*q(7)+8*ek*q(6)*q(4)*q(7)-(4*p1*ek*q(6)**2*q(3)*q(7))-(8*
. p1*ek*q(3)*q(7))-(24*p1**3*q(3)*q(7))+4*ek*q(6)**2*q(2)*q(7)+12*p1**2*q(6)
. **2*q(2)*q(7)+8*ek*q(2)*q(7)+4*p1**2*q(2)*q(7)+4*p1*ek*q(6)*q(1)*q(7)-(1
. 12*p1**3*q(6)*q(1)*q(7))+30*p1**3*q(6)**2+20*p1*ek*q(4)*q(6)+64*p1**3*q(4
. )*q(6)+20*p1**2*ek*q(1)*q(6)+4*p1**4*q(1)*q(6)-(4*p1*ek*q(3)*q(5))+4*
. ek*q(2)*q(5)+33*p1*ek*q(4)**2+46*p1**2*ek*q(1)*q(4)-(64*p1**4*q(1)*q(4)
. )+7*p1**3*ek*q(3)**2-(10*p1**2*ek*q(2)*q(3))+24*p1**4*q(2)*q(3)+3*p1*ek*
. q(2)**2-(14*p1**3*q(2)**2)+13*-1**3*ek*q(1)**2-(34*p1**5*q(1)**2))/6.0
bmn2(4,6)=T0+p1**2*ek**3*re*(3*ek*q(7)**2+24*p1**2*q(7)**2+16*p1**2
. *q(6)**2*q(5)*q(7)+6*ek*q(5)*q(7)+32*p1**2*q(5)*q(7)-(16*p1**3*q(2)*q(7
. )+9*p1**2*ek*q(6)**2+72*p1**4*q(6)**2+2+18*p1**2*ek*q(4)*q(6)+96*p1**4*
. q(4)*q(6)-(48*p1**5*q(1)*q(6))+3*ek*q(5)**2+8*p1**2*q(5)**2-(16*p1**3
. *q(2)*q(5))+9*p1**2*ek*q(4)**2+24*p1**4*q(4)**2-(48*p1**5*q(1)*q(4)))
. /6.0+p1*de*(6*q(6)*q(6)*q(7)+7*q(6)*q(4)*q(7)-(3*p1*q(6)**2*q(3)*q(7
. ))-(3*p1*q(3)*q(7))+3*q(6)**2*q(2)*q(7)+3*q(2)*q(7)+4*p1*q(6)*q(1)*q(7)

```

```

. +7*p1*q(4)*q(6)+7*p1**2*q(1)*q(6)+12*p1*q(4)**2+17*p1**2*q(1)*q(4)+4*
. p1**3*q(3)**2-(5*p1**2*q(2)*q(3))+p1*q(2)**2+5*p1**3*q(1)**2)/3.0
T0=p1**2*ek*q(22*ek*q(6)**2*q(6)*q(7)+36*ek*q(6)*q(7)+8*p1**2*q(6)*
. q(7)+28*ek*q(6)**2*q(4)*q(7)+50*ek*q(4)*q(7)+6*p1*ek*q(6)**2*q(1)*q(7)+14
. *p1*ek*q(1)*q(7)-(8*p1**3*q(1)*q(7))+6*p1*ek*q(6)*q(6)**2+13*ek*q(6)**2*q(5
. )*q(6)+21*ek*q(5)*q(6)+6*p1*ek*q(6)*q(4)*q(6)-(4*p1**2*ek*q(6)**2*q(3)*q(6
. ))-(30*p1**2*ek*q(3)*q(6))+6*p1*ek*q(6)**2*q(2)*q(6)+15*p1*ek*q(2)*q(6)-(8
. *p1**3*q(2)*q(6))+20*ek*q(6)**2*q(4)*q(5)+28*ek*q(4)*q(5)+7*p1*ek*q(6)**2
. *q(1)*q(5)+7*p1*ek*q(1)*q(5)-(4*p1**2*ek*q(6)**2*q(3)*q(4))-(32*p1**2*ek*
. q(3)*q(4))+6*p1*ek*q(6)**2*q(2)*q(4)+10*p1*ek*q(2)*q(4)-(2*p1**3*ek*q(1)*
. q(3))-(5*p1**2*ek*q(1)*q(2))+8*p1**4*q(1)*q(2))/3.0
T0=T0+2.0/3.0*p1**2*ek**2*q(1)*q(2)*ek**2*q(6)*q(7)+7*ek*q(6)*q(7)+
. 16*p1**2*q(6)*q(7)+2*ek*q(6)**2*q(4)*q(7)+10*ek*q(4)*q(7)+4*p1**2*q(4)*
. q(7)+3*p1*ek*q(1)*q(7)-(12*p1**3*q(1)*q(7))+ek*q(6)**2*q(5)*q(6)+6*p1**2
. *q(6)**2*q(5)*q(6)+6*p1*ek*q(5)*q(6)+6*p1**2*q(5)*q(6)-(4*p1**2*ek*q(3)*q(
. 6))+3*p1*ek*q(2)*q(6)-(10*p1**3*q(2)*q(6))+ek*q(6)**2*q(4)*q(5)+9*ek*q(4
. )*q(5)-(6*p1**3*q(6)**2*q(1)*q(5))+3*p1*ek*q(1)*q(5)-(6*p1**3*q(1)*q(5)
. )-(4*p1**2*ek*q(3)*q(4))+3*p1*ek*q(2)*q(4)-(4*p1**3*q(2)*q(4))+6*p1**4*
. q(1)*q(2))
T1=fe*(12*p1**2*q(6)**2*q(6)*q(7)+10*p1**2*q(6)*q(7)+2*ek*q(6)**2*q(4
. )*q(7)+2*ek*q(4)*q(7)+8*p1**2*q(4)*q(7)+2*p1*ek*q(6)**2*q(1)*q(7)-(12*p1
. **3*q(6)**2*q(1)*q(7))+2*p1*ek*q(1)*q(7)-(2*p1**3*q(1)*q(7))+2*p1*ek*q(6)*
. q(6)**2+6*p1**3*q(6)*q(6)**2+4*p1*ek*q(6)*q(4)*q(6)-(2*p1**2*ek*q(6)**2*q(3
. )*q(6))-(4*p1**2*ek*q(3)*q(6))-(12*p1**4*q(3)*q(6))+2*p1*ek*q(6)**2*q(2)*
. q(6)+6*p1**3*q(6)**2*q(2)*q(6)+4*p1*ek*q(2)*q(6)+2*p1**3*q(2)*q(6)+2*p1
. **2*ek*q(6)*q(1)*q(6)-(6*p1**4*q(6)*q(1)*q(6))+ek*q(6)**2*q(4)*q(5)+ek*q(4
. )*q(5)+p1*ek*q(6)**2*q(1)*q(5)+p1*ek*q(1)*q(5)-(4*p1**2*ek*q(6)**2*q(3)*q(4
. )-(10*p1**2*ek*q(3)*q(4))+4*p1*ek*q(6)**2*q(2)*q(4)+9*p1*ek*q(2)*q(4)-(8*
. p1**3*q(2)*q(4))-(2*p1**3*ek*q(6)**2*q(1)*q(3))-(6*p1**3*ek*q(1)*q(3))+
. 12*p1**5*q(1)*q(3)+2*p1**2*ek*q(6)**2*q(1)*q(2)-(6*p1**4*q(6)**2*q(1)*q(2
. ))+5*p1**2*ek*q(1)*q(2)-(10*p1**4*q(1)*q(2)))
T0=T0+ek*q(20*p1**2*q(6)**2*q(6)*q(7)+28*p1**2*q(6)*q(7)+16*p1**2*
. q(6)**2*q(4)*q(7)+ek*q(4)*q(7)+36*p1**2*q(4)*q(7)-(4*p1**3*q(6)**2*q(1)*
. q(7))+p1*ek*q(1)*q(7)+8*p1**3*q(1)*q(7)+10*p1**3*q(6)*q(6)**2+8*p1**2*q(6)
. **2*q(5)*q(6)+8*p1**2*q(5)*q(6)+8*p1**3*q(6)*q(4)*q(6)-(4*p1**4*q(6)**2*
. q(3)*q(6))-(3*p1**2*ek*q(3)*q(6))-(24*p1**4*q(3)*q(6))+10*p1**3*q(6)**2
. *q(2)*q(6)+3*p1*ek*q(2)*q(6)+4*p1**3*q(2)*q(6)-(2*p1**4*q(6)*q(1)*q(6))
. +8*p1**2*q(6)**2*q(4)*q(5)+ek*q(4)*q(5)+8*p1**2*q(4)*q(5)+p1*ek*q(1)*q(5
. )-(6*p1**2*ek*q(3)*q(4))-(16*p1**4*q(3)*q(4))+8*p1**3*q(6)**2*q(2)*q(4)
. +6*p1*ek*q(2)*q(4)-(12*p1**3*q(2)*q(4))+4*p1**5*q(6)**2*q(1)*q(3)-(3*p1
. **3*ek*q(1)*q(3))+8*p1**5*q(1)*q(3)-(2*p1**4*q(6)**2*q(1)*q(2))+3*p1**2
. *ek*q(1)*q(2)-(16*p1**4*q(1)*q(2)))/3.0+T1/3.0
bmn2(4,7)=T0+p1**2*ek**3*re*(3*ek*q(6)*q(7)+24*p1**2*q(6)*q(7)+3*ek
. *q(4)*q(7)+16*p1**2*q(4)*q(7)-(8*p1**3*q(1)*q(7))+8*p1**2*q(6)**2*q(5)
. *q(6)+3*ek*q(5)*q(6)+16*p1**2*q(5)*q(6)-(8*p1**3*q(2)*q(6))+8*p1**2*
. q(6)**2*q(4)*q(5)+3*ek*q(4)*q(5)+8*p1**2*q(4)*q(5)-(8*p1**3*q(1)*q(5))
. -(8*p1**3*q(2)*q(4)))/3.0+de*(2*q(6)**2*q(4)*q(7)+q(4)*q(7)+2*p1
. *q(6)**2*q(1)*q(7)+p1*q(1)*q(7)+3*p1*q(6)*q(6)**2+7*p1*q(6)*q(4)*q(6)-(3*p1
. **2*q(6)**2*q(3)*q(6))-(3*p1**2*q(3)*q(6))+3*p1*q(6)**2*q(2)*q(6)+3*p1*q
. (2)*q(6)+4*p1**2*q(6)*q(1)*q(6)-(6*p1**2*q(6)**2*q(3)*q(4))-(8*p1**2*q(3
. )*q(4))+7*p1*q(6)**2*q(2)*q(4)+7*p1*q(2)*q(4)-(3*p1**3*q(6)**2*q(1)*q(3)
. )-(5*p1**3*q(1)*q(3))+4*p1**2*q(6)**2*q(1)*q(2)+4*p1**2*q(1)*q(2))/3.0
. +4*p1**4*ek**4*te*(q(6)+q(4))*(q(7)+q(5))
T0=2*p1**2*ek**4*te*(3*(q(7)+q(5))**2+p1**2*(q(6)+q(4))**2)+2.0/
. 3.0*ek**2*el*(6*p1**2*q(6)**4*q(7)**2+3*ek*q(6)**2*q(7)**2+12*p1**2*
. q(6)**2*q(7)**2+3*ek*q(7)**2+8*p1**2*q(7)**2+3*ek*q(6)**2*q(5)**q(7)+3*ek*
. q(5)*q(7)-(12*p1**2*ek*q(3)*q(7))-(12*p1**3*q(6)**2*q(2)*q(7))+9*p1*ek*q
. (2)*q(7)-(16*p1**3*q(2)*q(7))+p1**2*ek*q(6)**2+2*p1**4*q(6)**2+5*p1**2
. *ek*q(4)*q(6)+3*p1**3*ek*q(1)*q(6)-(4*p1**5*q(1)*q(6))-(12*p1**2*ek*q(3
. )*q(5))+9*p1*ek*q(2)*q(5)+4*p1**2*ek*q(4)**2+3*p1**3*ek*q(1)*q(4)+8*p1**2
. *q(4)*q(2)**2+2*p1**6*q(1)**2)
bmn2(5,5)=T0+ek**2*ej*(3*q(6)**4*q(7)**2+8*q(6)**2*q(7)**2+4*q(7)**2
. +2+6*p1*q(6)*q(7)-(24*p1**2*q(6)**2*q(3)*q(7))-(28*p1**2*q(3)*q(7))+
. 20*p1*q(6)**2*q(2)*q(7)+20*p1*q(2)*q(7)+p1**2*q(6)**2+8*p1**2*q(4)*q(6)
. +6*p1**3*q(1)*q(6)+4*p1**2*q(4)**2+16*p1**4*q(3)**2-(4*p1**3*q(2)*q(3
. ))-(8*p1**2*q(2)**2)-(3*p1**4*q(1)**2))/3.0+ek**3*re*(32*p1**2*q(6)
. **2*q(7)**2+9*ek*q(7)**2+48*p1**2*q(7)**2+48*p1**2*q(6)**2*q(5)*q(7)+18
. *ek*q(5)*q(7)+48*p1**2*q(5)*q(7)-(48*p1**3*q(2)*q(7))+3*p1**2*ek*q(6
. )**2+16*p1**4*q(6)**2+6*p1**2*ek*q(4)*q(6)+16*p1**4*q(4)*q(6)-(16*p1**2
. *5*q(1)*q(6))+9*ek*q(5)**2-(48*p1**3*q(2)*q(5))+3*p1**2*ek*q(4)**2-(16
. *p1**5*q(1)*q(4))/6.0+ek**2*he*(q(4)**2+2*p1*q(1)*q(4)+7*p1**2*q
. (3)**2-(14*p1*q(2)*q(3))+7*q(2)**2+p1**2*q(1)**2)/6.0

```

```

T0=p1*ek*he*(6*q(6)**3*q(7)**2+6*q(6)*q(7)**2+2*p1*q(6)**2*q(6)*q(7)+2*
. p1*q(6)*q(7)+8*p1*q(6)**2*q(4)*q(7)+8*p1*q(4)*q(7)-(4*p1**2*q(6)*q(3)*q(7)
. )-(2*p1*q(6)*q(2)*q(7))+6*p1**2*q(6)**2*q(1)*q(7)+6*p1**2*q(1)*q(7)-(10
. *p1**3*q(3)*q(6))+8*p1**2*q(2)*q(6)-(3*p1*ek*q(3)*q(4))-(8*p1**3*q(3)*
. q(4))+3*ek*q(2)*q(4)-(3*p1**2*ek*q(1)*q(3))+2*p1**4*q(1)*q(3)+3*p1*ek*q
. (1)*q(2)-(8*p1**3*q(1)*q(2)))/3.0
T0=T0+p1*ek*ej*(6*ek*q(6)*q(7)**2+6*p1*ek*q(6)**2*q(6)*q(7)+8*p1**3*q(6)**2
. 2*q(6)*q(7)+8*p1*ek*q(6)*q(7)+8*p1**3*q(6)*q(7)+6*ek*q(6)*q(5)*q(7)+13*
. p1*ek*q(6)**2*q(4)*q(7)+21*p1*ek*q(4)*q(7)+7*p1**2*ek*q(6)**2*q(1)*q(7)-(8*
. p1**4*q(7)**2*q(1)*q(7))+13*p1**2*ek*q(1)*q(7)-(8*p1**4*q(1)*q(7))+2*p1*
. ek*q(5)*q(6)-(20*p1**3*q(3)*q(6))+14*p1**2*ek*q(2)*q(6)-(8*p1**4*q(2)
. )*q(6))+8*p1*ek*q(4)*q(5)+6*p1**2*ek*q(1)*q(5)-(22*p1**3*ek*q(3)*q(4))+
. 9*p1**2*ek*q(2)*q(4)-(2*p1**4*ek*q(1)*q(3))-(5*p1**3*ek*q(1)*q(2))+8*p1
. **5*q(1)*q(2))/3.0
T0=T0+2.0/3.0*p1**2*ek**2*el*(ek*q(6)**2*q(6)*q(7)+12*p1**2*q(6)**2*q(
. 6)*q(7)+3*ek*q(6)*q(7)+16*p1**2*q(6)*q(7)+ek*q(6)**2*q(4)*q(7)+6*p1**2*
. q(6)**2*q(4)*q(7)+6*ek*q(4)*q(7)+6*p1**2*q(4)*q(7)-(6*p1**3*q(6)**2*q(1)
. *q(7))+3*p1*ek*q(1)*q(7)-(10*p1**3*q(1)*q(7))+2*ek*q(5)*q(6)+4*p1**2*q
. (5)*q(6)-(4*p1**2*ek*q(3)*q(6))+3*p1*ek*q(2)*q(6)-(12*p1**3*q(2)*q(6))
. +5*ek*q(4)*q(5)+3*p1*ek*q(1)*q(5)-(4*p1**3*q(1)*q(5))-(4*p1**2*ek*q(3)*
. q(4))+3*p1*ek*q(2)*q(4)-(6*p1**3*q(2)*q(4))+6*p1**4*q(1)*q(2))+p1**2*ek
. **3*re*(8*p1**2*q(6)**2*q(6)*q(7)+3*ek*q(6)*q(7)+24*p1**2*q(6)*q(7)
. )+8*p1**2*q(6)**2*q(4)*q(7)+3*ek*q(4)*q(7)+16*p1**2*q(4)*q(7)-(8*p1**3*
. q(1)*q(7))+3*ek*q(5)*q(6)+16*p1**2*q(5)*q(6)-(8*p1**3*q(2)*q(6))+3*ek
. *q(4)*q(5)+8*p1**2*q(4)*q(5)-(8*p1**3*q(1)*q(5))-(8*p1**3*q(2)*q(4))
. )/3.0
bmn2(5,6)=T0-(2.0/3.0*ek*fe*(p1*q(3)-q(2))*(q(6)*q(7)+p1*(q(4)+p1*q(
. 1)))+4*p1**4*ek*4*te*(q(6)+q(4))*(q(7)+q(5))
T0=9*ek*q(6)**4*q(7)**2+24*ek*q(6)**2*q(7)**2+12*p1**2*q(6)**2*q(7)**2+12*ek
. *q(7)**2+12*p1**2*q(7)**2+12*p1*ek*q(6)*q(7)+6*ek*q(6)**4*q(5)*q(7)+
. 16*ek*q(6)**2*q(5)*q(7)+8*ek*q(5)*q(7)-(40*p1**2*ek*q(6)**2*q(3)*q(7))-
. 56*p1**2*ek*q(3)*q(7))+32*p1*ek*q(6)**2*q(2)*q(7)-(16*p1**3*q(6)**2*q(2)*q
. (7))+40*p1*ek*q(2)*q(7)-(24*p1**3*q(2)*q(7))+3*p1**2*ek*q(6)**2*q(6)**2+
. 4*p1**4*q(6)**2*q(6)**2+4*p1**2*ek*q(6)**2+4*p1**4*q(6)**2+6*p1*ek*q(6)*q(5
. )*q(6)+13*p1**2*ek*q(6)**2*q(4)*q(6)+21*p1**2*ek*q(4)*q(6)+7*p1**3*ek*q(6)
. **2*q(1)*q(6)-(8*p1**5*q(6)**2*q(1)*q(6))+13*p1**3*ek*q(1)*q(6)-(8*p1**5
. *q(1)*q(6)-(24*p1**2*ek*q(6)**2*q(3)*q(5))-(28*p1**2*ek*q(3)*q(5))+20
. *p1*ek*q(6)**2*q(2)*q(5)+20*p1*ek*q(2)*q(5)+10*p1**2*ek*q(6)**2*q(4)**2+14*
. p1**2*ek*q(4)**2+7*p1**3*ek*q(6)**2*q(1)*q(4)+7*p1**3*ek*q(1)*q(4)+16*p1**
. 4*ek*q(3)**2
T0=2*p1**2*ek**4*te*(3*(q(7)+q(5))**2+p1**2*(q(6)+q(4))**2)+ek*ej
. *(T0-(4*p1**3*ek*q(2)*q(3))+4*p1**4*q(6)**2*q(2)**2-(8*p1**2*ek*q(2)
. **2)+12*p1**4*q(2)**2+4*p1**6*q(6)**2*q(1)**2-(3*p1**4*ek*q(1)**2)+4*p1
. **6*q(1)**2)/3.0
T1=9*ek*q(6)**2*q(7)**2+36*p1**2*q(6)**2*q(7)**2+9*ek*q(7)**2+48*p1**2*q(7)
. **2+24*p1**2*q(6)**4*q(5)*q(7)+12*ek*q(6)**2*q(5)*q(7)+48*p1**2*q(6)**2*q
. (5)*q(7)+12*ek*q(5)*q(7)+32*p1**2*q(5)*q(7)-(24*p1**2*ek*q(3)*q(7))-
. 24*p1**3*q(6)**2*q(2)*q(7))+18*p1*ek*q(2)*q(7)-(64*p1**3*q(2)*q(7))+p1**
. 2*ek*q(6)**2*q(6)**2+12*p1**4*q(6)**2*q(6)**2+3*p1**2*ek*q(6)**2+16*p1**4*
. q(6)**2+2*p1**2*ek*q(6)**2*q(4)*q(6)+12*p1**4*q(6)**2*q(4)*q(6)+12*p1**2*
. ek*q(4)*q(6)+12*p1**4*q(4)*q(6)-(12*p1**5*q(6)**2*q(1)*q(6))+6*p1**3*ek*
. q(1)*q(6)-(20*p1**5*q(1)*q(6))+3*ek*q(6)**2*q(5)**2+3*ek*q(5)**2-(24*p1
. **2*ek*q(3)*q(5))-(24*p1**3*q(6)**2*q(2)*q(5))+18*p1*ek*q(2)*q(5)-(32*p1
. **3*q(2)*q(5))+p1**2*ek*q(6)**2*q(4)**2+9*p1**2*ek*q(4)**2-(12*p1**5*q(6)
. **2*q(1)*q(4))+6*p1**3*ek*q(1)*q(4)-(12*p1**5*q(1)*q(4))+16*p1**4*q(2)
. )**2
T0=T0+ek*he*(12*q(6)**4*q(7)**2+18*q(6)**2*q(7)**2+6*q(7)**2+24*p1*
. q(6)**3*q(6)*q(7)+24*p1*q(6)*q(6)*q(7)-(16*p1**2*q(6)**4*q(3)*q(7))-(64*p1
. **2*q(6)**2*q(3)*q(7))-(52*p1**2*q(3)*q(7))+24*p1*q(6)**4*q(2)*q(7)+56*
. p1*q(6)**2*q(2)*q(7)+(40*p1*q(2)*q(7))+2*p1**2*q(6)**2*q(6)**2+2*p1**2*q(6)
. **2+16*p1**2*q(6)**2*q(4)*q(6)+16*p1**2*q(4)*q(6)-(8*p1**3*q(6)*q(3)*q(6
. ))-(4*p1**2*q(6)*q(2)*q(6))+12*p1**3*q(6)**2*q(1)*q(6)+12*p1**3*q(1)*q(6
. )+8*p1**2*q(6)**2*q(4)**2+2*p1*ek*q(4)**2+8*p1**2*q(4)**2+2*p1*ek*q(1)*q(4)+8
. *p1**4*q(6)**2*q(3)**2+7*p1**2*ek*q(3)**2+24*p1**4*q(3)**2-(14*p1*ek*q(2)
. *q(3))+4*p1**3*q(2)*q(3)-(10*p1**2*q(6)**2*q(2)**2)+7*ek*q(2)**2-(22*p1
. **2*q(2)**2)-(6*p1**4*q(6)**2*q(1)**2)+p1**2*ek*q(1)**2-(6*p1**4*q(1)**2
. 2))/6.0+ek**2*el*(T1+4*p1**6*q(1)**2)/3.0
bmn2(5,7)=T0+ek**3*re*(24*p1**2*q(6)**2*q(7)**2+9*ek*q(7)**2+72*p1**
. 2*q(7)**2+64*p1**2*q(6)**2*q(5)*q(7)+18*ek*q(5)*q(7)+96*p1**2*q(5)*q(7)
. )-(48*p1**3*q(2)*q(7))+8*p1**4*q(6)**2*q(6)**2+3*p1**2*ek*q(6)**2+24*p1
. **4*q(6)**2+16*p1**4*q(6)**2*q(4)*q(6)+6*p1**2*ek*q(4)*q(6)+32*p1**4*q(
. 4)*q(6)-(16*p1**5*q(1)*q(6))+24*p1**2*q(6)**2*q(5)**2+9*ek*q(5)**2+24*

```

```

. p1**2*q(5)**2-(48*p1**3*q(2)*q(5))+8*p1**4*q(6)**2+3*p1**2*ek*q(
. 4)**2+8*p1**4*q(4)**2-(16*p1**5*q(1)*q(4)))/6.0-(1.0/6.0*ek*fe*(

. 4*p1*q(6)*q(3)*q(6)-(4*q(6)*q(2)*q(6))-(q(6)**2*q(4)**2)-q(4)**2-(2*p1*q(6)
. **2*q(1)*q(4))-(2*p1*q(1)*q(4))-(5*p1**2*q(6)**2*q(3)**2)-(5*p1**2*q(3
. )**2)+10*p1*q(6)**2*q(2)*q(3)+10*p1*q(2)*q(3)-(5*q(6)**2*q(2)**2)-(5*q(
. 2)**2)-(p1**2*(q(6)**2+1)*q(1)**2)))
T0=2*p1**4*ek**4*te*((q(7)+q(5))**2+3*p1**2*(q(6)+q(4))**2)+p1**2*
. he*(12*ek*q(6)**2*q(7)**2+8*ek*q(7)**2+12*p1**2*q(7)**2+72*p1*ek*q(6)*
. q(6)*q(7)+4*ek*q(6)**2*q(5)*q(7)+4*ek*q(5)*q(7)+40*p1*ek*q(6)*q(4)*q(7)-(

. 16*p1**2*ek*q(6)**2*q(3)*q(7))-(52*p1**2*ek*q(3)*q(7))+24*p1*ek*q(6)**2*q(2
. )*q(7)+40*p1*ek*q(2)*q(7)-(24*p1**3*q(2)*q(7))-(8*p1**2*ek*q(6)*q(1)*q(7
. ))+24*p1**2*ek*q(6)**2+36*p1**4*q(6)**2+168*p1**2*ek*q(4)*q(6)+120*p1**3
. *ek*q(1)*q(6)-(72*p1**5*q(1)*q(6))-(20*p1**2*ek*q(3)*q(5))+16*p1*ek*q(1
. 2)*q(5)+7*ek**2*q(4)**2+100*p1**2*ek*q(4)**2+14*p1*ek**2*q(1)*q(4)+32*
. p1**3*ek*q(1)*q(4)+p1**2*ek**2*q(3)**2+16*p1**4*ek*q(3)**2-(2*p1*ek**2*q(
. 2)*q(3))+ek**2*q(2)**2-(12*p1**2*ek*q(2)**2)+12*p1**4*q(2)**2+7*p1**2*
. ek**2*q(1)**2-(44*p1**4*ek*q(1)**2)+36*p1**6*q(1)**2)/6.0
T0=T0+p1**2*ek*ej*(8*ek*q(6)**2*q(7)**2+8*ek*q(7)**2+16*p1**2*q(7)**2
. +18*p1*ek*q(6)*q(7)+6*ek*q(6)**2*q(5)*q(7)+8*p1**2*q(6)**2*q(5)*q(7)+8
. *ek*q(5)*q(7)+8*p1**2*q(5)*q(7)+12*p1*ek*q(6)*q(4)*q(7)-(4*p1**2*ek*q(6)**2
. *q(3)*q(7))-(28*p1**2*ek*q(3)*q(7))+6*p1*ek*q(6)**2*q(2)*q(7)+20*p1*ek*q(
. 2)*q(7)-(24*p1**3*q(2)*q(7))+24*p1**2*ek*q(6)**2+48*p1**4*q(6)**2+108
. *p1**2*ek*q(4)*q(6)+24*p1**4*q(4)*q(6)+60*p1**3*ek*q(1)*q(6)-(72*p1**5*
. q(1)*q(6))+ek*q(5)**2-(20*p1**2*ek*q(3)*q(5))+14*p1*ek*q(2)*q(5)-(8*p1
. **3*q(2)*q(5))+76*p1**2*ek*q(4)**2+44*p1**3*ek*q(1)*q(4)-(24*p1**5*q(1
. )*q(4))+4*p1**4*ek*q(3)**2-(3*p1**2*ek*q(2)**2)+8*p1**4*q(2)**2-(8*p1**4
. *ek*q(1)**2)+24*p1**6*q(1)**2)/3.0
T0=T0+2.0/3.0*p1**2*ek**2*ek**2*el*(ek*q(6)**2*q(7)**2+2*ek*q(7)**2+16*p1**2
. 2*q(7)**2+ek*q(6)**2*q(5)*q(7)+12*p1**2*q(6)**2*q(5)*q(7)+3*ek*q(5)*q(7)
. +16*p1**2*q(5)*q(7)-(4*p1**2*ek*q(3)*q(7))+3*p1*ek*q(2)*q(7)-(16*p1**3*
. q(2)*q(7))+6*p1**2*ek*q(6)**2+48*p1**4*q(6)**2+21*p1**2*ek*q(4)*q(6)+
. 48*p1**4*q(4)*q(6)+9*p1**3*ek*q(1)*q(6)-(48*p1**5*q(1)*q(6))+ek*q(5)**2
. +2+2*p1**2*q(5)**2-(4*p1**2*ek*q(3)*q(5))+3*p1*ek*q(2)*q(5)-(12*p1**3*q(
. 2)*q(5))+15*p1**2*ek*q(4)**2+8*p1**4*q(4)**2+9*p1**3*ek*q(1)*q(4)-(32*
. p1**5*q(1)*q(4))+2*p1**4*q(2)**2+8*p1**6*q(1)**2)
T0=T0+p1*fe*(12*p1*q(6)**2*q(7)**2+3*p1*q(7)**2+42*p1**2*q(6)*q(6)*q(7)
. +8*ek*q(6)*q(4)*q(7)+24*p1**2*q(6)*q(4)*q(7)-(8*p1**3*q(6)**2*q(3)*q(7))-
. (20*p1**3*q(3)*q(7))+14*p1**2*q(6)**2*q(2)*q(7)+14*p1**2*q(2)*q(7)+8*p1
. *ek*q(6)*q(1)*q(7)-(4*p1**3*q(6)*q(1)*q(7))+9*p1**3*q(6)**2+60*p1**3*q(4)
. *q(6)+42*p1**4*q(1)*q(6)+10*p1*ek*q(4)**2+32*p1**3*q(4)**2+20*p1**2*ek*
. q(1)*q(4)+4*p1**4*q(1)*q(4)+2*p1**3*ek*q(3)**2+8*p1**5*q(3)**2-(4*p1**2
. *ek*q(2)*q(3))+4*p1**4*q(2)*q(3)+2*p1*ek*q(2)**2-(9*p1**3*q(2)**2)+10
. *p1**3*ek*q(1)**2-(19*p1**5*q(1)**2))/6.0+de*(6*q(6)**2*q(7)**2+
. 12*p1*q(6)*q(4)*q(7)+12*p1**2*q(6)*q(1)*q(7)+7*p1**2*q(4)**2+14*p1**3*q(1
. )*q(4)+p1**4*q(3)**2-(2*p1**3*q(2)*q(3))+p1**2*q(2)**2+7*p1**4*q(1)**2
. 2)/6.0
bmn2(6,6)=T0+p1**2*ek**3*re*(3*ek*q(7)**2+32*p1**2*q(7)**2+16*p1**2
. *q(6)**2*q(5)*q(7)+6*ek*q(5)*q(7)+48*p1**2*q(5)*q(7)-(16*p1**3*q(2)*q(7)
. 7))+9*p1**2*ek*q(6)**2+96*p1**4*q(6)**2+18*p1**2*ek*q(4)*q(6)+144*p1**4
. *q(4)*q(6)-(48*p1**5*q(1)*q(6))+3*ek*q(5)**2+16*p1**2*q(5)**2-(16*p1
. **3*q(2)*q(5))+9*p1**2*ek*q(4)**2+24*p1**4*q(4)**2-(48*p1**5*q(1)*q(4
. ))/6.0
T0=6*ek*q(6)**3*q(7)**2+18*ek*q(6)*q(7)**2+12*p1*ek*q(6)**2*q(6)*q(7)+8*p1*ek*
. q(6)*q(7)+12*p1**3*q(6)*q(7)+12*ek*q(6)**3*q(5)*q(7)+12*ek*q(6)*q(5)*q(7)
. +20*p1*ek*q(6)**2*q(4)*q(7)+28*p1*ek*q(4)*q(7)-(8*p1**2*ek*q(6)*q(3)*q(7))
. -(4*p1*ek*q(6)*q(2)*q(7))+8*p1**2*ek*q(6)**2*q(1)*q(7)+20*p1**2*ek*q(1)*q(7)
. -(12*p1**4*q(1)*q(7))+18*p1**2*ek*q(6)*q(6)**2+2*p1*ek*q(6)**2*q(5)*q(6)+
. 2*p1*ek*q(5)*q(6)+20*p1**2*ek*q(6)*q(4)*q(6)-(8*p1**3*ek*q(6)**2*q(3)*q(6))
. -(26*p1**3*ek*q(3)*q(6))+12*p1**2*ek*q(6)**2*q(2)*q(6)+20*p1**2*ek*q(2)*q
. (6)-(12*p1**4*q(2)*q(6))-(4*p1**3*ek*q(6)*q(1)*q(6))+8*p1*ek*q(6)**2*q(4)*
. q(5)+8*p1*ek*q(4)*q(5)-(4*p1**2*ek*q(6)*q(3)*q(5))-(2*p1*ek*q(6)*q(2)*q(5))
. +6*p1**2*ek*q(6)**2*q(1)*q(5)+6*p1**2*ek*q(1)*q(5)+4*p1**2*ek*q(6)*q(4)**2
. -(4*p1**3*ek*q(6)**2*q(3)*q(4))-(3*p1*ek**2*q(3)*q(4))-(24*p1**3*ek*q(3)*q
. (4))+10*p1**2*ek*q(6)**2*q(2)*q(4)
T1=18*p1*q(6)**3*q(7)**2+21*p1*q(6)*q(7)**2+24*p1**2*q(6)**2*q(6)*q(7)+6*p1
. **2*q(6)*q(7)+24*p1**2*q(6)**2*q(4)*q(7)+20*p1**2*q(4)*q(7)-(8*p1*ek*q(6)
. *q(3)*q(7))-(24*p1**3*q(6)*q(3)*q(7))+12*p1**2*q(6)**3*q(2)*q(7)+8*ek*q(6)
. *q(2)*q(7)-(4*p1**2*q(6)*q(2)*q(7))+12*p1**3*q(6)**2*q(1)*q(7)+14*p1**3*
. q(1)*q(7)+21*p1**3*q(6)*q(6)**2+8*p1*ek*q(6)*q(4)*q(6)+24*p1**3*q(6)*q(4)*q
. (6)-(8*p1**4*q(6)**2*q(3)*q(6))-(20*p1**4*q(3)*q(6))+14*p1**3*q(6)**2*q(1)*q
. (2)*q(6)+14*p1**3*q(2)*q(6)+8*p1**2*ek*q(6)*q(1)*q(6)-(4*p1**4*q(6)*q(1)*q
. (6))-(4*p1*ek*q(6)*q(3)*q(5))+4*ek*q(6)*q(2)*q(5)+4*p1*ek*q(6)*q(4)**2-(4*p1

```

```

. **2*ek*q(6)**2*q(3)*q(4))-(8*p1**2*ek*q(3)*q(4))-(24*p1**4*q(3)*q(4))+4
. *p1*ek*q(6)**2*q(2)*q(4)+12*p1**3*q(6)**2*q(2)*q(4)+8*p1*ek*q(2)*q(4)+4*p1
. **3*q(2)*q(4)+4*p1**2*ek*q(6)*q(1)*q(4)-(12*p1**4*q(6)*q(1)*q(4))+12*p1**
. 4*q(6)*q(2)*q(3)-(4*p1**3*ek*q(6)**2*q(1)*q(3))
T2=9*ek*q(6)*q(7)**2+16*p1*ek*q(6)**2*q(6)*q(7)+16*p1*ek*q(6)*q(7)+32*p1**3*
. q(6)*q(7)+12*ek*q(6)*q(5)*q(7)+22*p1*ek*q(6)**2*q(4)*q(7)+36*p1*ek*q(4)*q(
. 7)+8*p1**3*q(4)*q(7)+6*p1**2*ek*q(6)**2*q(1)*q(7)+20*p1**2*ek*q(1)*q(7)-
. (24*p1**4*q(1)*q(7))+9*p1**2*ek*q(6)*q(6)**2+6*p1*ek*q(6)**2*q(5)*q(6)+8*p1
. **3*q(6)**2*q(5)*q(6)+8*p1*ek*q(5)*q(6)+8*p1**3*q(5)*q(6)+12*p1**2*ek*q(6)
. *q(4)*q(6)-(4*p1**3*ek*q(6)**2*q(3)*q(6))-(28*p1**3*ek*q(3)*q(6))+6*p1**
. 2*ek*q(6)**2*q(2)*q(6)+20*p1**2*ek*q(2)*q(6)-(24*p1**4*q(2)*q(6))+3*ek*
. q(6)*q(5)**2+13*p1*ek*q(6)**2*q(4)*q(5)+21*p1*ek*q(4)*q(5)+7*p1**2*ek*q(6)**2
. *q(1)*q(5)-(8*p1**4*q(6)**2*q(1)*q(5))+13*p1**2*ek*q(1)*q(5)-(8*p1**4*q
. (1)*q(5))+3*p1**2*ek*q(6)*q(4)**2-(4*p1**3*ek*q(6)**2*q(3)*q(4))-(30*p1**3
. *ek*q(3)*q(4))+6*p1**2*ek*q(6)**2*q(2)*q(4)+15*p1**2*ek*q(2)*q(4)-(8*p1**
. 4*q(2)*q(4))-(2*p1**4*ek*q(1)*q(3))
T0=p1*he*(T0+3*ek**2*q(2)*q(4)+4*p1**2*ek*q(2)*q(4)-(2*p1**3*ek*q(6)*q
. (1)*q(4))+4*p1**4*ek*q(6)**2*q(1)*q(3)-(3*p1**2*ek**2*q(1)*q(3))+2*p1**4
. *ek*q(1)*q(3)-(2*p1**3*ek*q(6)**2*q(1)*q(2))+3*p1*ek**2*q(1)*q(2)-(16*p1
. **3*ek*q(1)*q(2))+12*p1**5*q(1)*q(2))/3.0+fe*(T1+8*p1**5*q(6)**2*q
. (1)*q(3)-(8*p1**3*ek*q(1)*q(3))-(4*p1**5*q(1)*q(3))-(5*p1**3*q(6)*q(2)
. **2)+4*p1**2*ek*q(6)**2*q(1)*q(2)-(2*p1**4*q(6)**2*q(1)*q(2))+8*p1**2*ek*q
. (1)*q(2)-(10*p1**4*q(1)*q(2))-(5*p1**5*q(6)*q(1)**2))/6.0+p1*ek**ej*
. (T2-(5*p1**3*ek*q(1)*q(2))+16*p1**5*q(1)*q(2))/3.0
T0=T0+2.0/3.0*p1**2*ek**2*el*(2*ek*q(6)**2*q(6)*q(7)+4*ek*q(6)*q(7)+
. 32*p1**2*q(6)*q(7)+2*ek*q(6)**2*q(4)*q(7)+7*ek*q(4)*q(7)+16*p1**2*q(4)*
. q(7)+3*p1*ek*q(1)*q(7)-(16*p1**3*q(1)*q(7))+ek*q(6)**2*q(5)*q(6)+12*p1**
. 2*q(6)**2*q(5)*q(6)+3*ek*q(5)*q(6)+16*p1**2*q(5)*q(6)-(4*p1**2*ek*q(3)*
. q(6))+3*p1*ek*q(2)*q(6)-(16*p1**3*q(2)*q(6))+ek*q(6)**2*q(4)*q(5)+6*p1**
. 2*q(6)**2*q(4)*q(5)+6*ek*q(4)*q(5)+6*p1**2*q(4)*q(5)-(6*p1**3*q(6)**2*q
. 1)*q(5))+3*p1*ek*q(1)*q(5)-(10*p1**3*q(1)*q(5))-(4*p1**2*ek*q(3)*q(4))
. +3*p1*ek*q(2)*q(4)-(10*p1**3*q(2)*q(4))+6*p1**4*q(1)*q(2))
bmn2(6,7)=T0+de*(12*q(6)**2*q(6)*q(7)-(12*p1*q(6)**3*q(3)*q(7))-
. 12*p1*q(6)*q(3)*q(7))+12*q(6)**3*q(2)*q(7)+12*q(6)*q(2)*q(7)+12*p1*q(6)*q(4)*
. )*q(6)+12*p1**2*q(6)*q(1)*q(6)+7*p1*q(6)*q(4)**2-(6*p1**2*q(6)**2*q(3)*q(4)
. )-(6*p1**2*q(3)*q(4))+6*p1*q(6)**2*q(2)*q(4)+6*p1*q(2)*q(4)+8*p1**2*q(6)
. *q(1)*q(4)+7*p1**3*q(6)*q(3)**2-(8*p1**2*q(6)*q(2)*q(3))-(6*p1**3*q(6)**2*
. q(1)*q(3))-(6*p1**3*q(1)*q(3))+p1*q(6)*q(2)**2+6*p1**2*q(6)**2*q(1)*q(2)
. +6*p1**2*q(1)*q(2)+p1**3*q(6)*q(1)**2)/6.0+p1**2*ek**3*re*(3*ek*q(6)
. *q(7)+32*p1**2*q(6)*q(7)+3*ek*q(4)*q(7)+24*p1**2*q(4)*q(7)-(8*p1**3*q
. (1)*q(7))+8*p1**2*q(6)**2*q(5)*q(6)+3*ek*q(5)*q(6)+24*p1**2*q(5)*q(6)-
. (8*p1**3*q(2)*q(6))+8*p1**2*q(6)**2*q(4)*q(5)+3*ek*q(4)*q(5)+16*p1**2*q
. (4)*q(5)-(8*p1**3*q(1)*q(5))-(8*p1**3*q(2)*q(4)))/3.0+4*p1**4*ek**4*te
. *(q(6)+q(4))*(q(7)+q(5))
T0=48*ek*q(6)**4*q(7)**2+72*ek*q(6)**2*q(7)**2+24*ek*q(7)**2+36*p1**2*q(7)
. **2+24*p1*ek*q(6)**3*q(6)*q(7)+72*p1*ek*q(6)*q(6)*q(7)+24*ek*q(6)**4*q(5)*q(
. 7)+36*ek*q(6)**2*q(5)*q(7)+12*ek*q(5)*q(7)-(144*p1**2*ek*q(6)**2*q(3)*q(7)
. )-(156*p1**2*ek*q(3)*q(7))+24*p1*ek*q(6)**4*q(2)*q(7)+120*p1*ek*q(6)**2*q(
. 2)*q(7)+120*p1*ek*q(2)*q(7)-(72*p1**3*q(2)*q(7))+12*p1**2*ek*q(6)**2*q(6)
. **2+8*p1**2*ek*q(6)**2+12*p1**4*q(6)**2+24*p1*ek*q(6)**3*q(5)*q(6)+24*p1
. *ek*q(6)*q(5)*q(6)+40*p1**2*ek*q(6)**2*q(4)*q(6)+56*p1**2*ek*q(4)*q(6)-(16
. *p1**3*ek*q(6)*q(3)*q(6))-(8*p1**2*ek*q(6)*q(2)*q(6))+16*p1**3*ek*q(6)**2*q(1
. )*q(6)+40*p1**3*ek*q(1)*q(6)-(24*p1**5*q(1)*q(6))-(16*p1**2*ek*q(6)**4*q
. (3)*q(5))-(64*p1**2*ek*q(6)**2*q(3)*q(5))-(52*p1**2*ek*q(3)*q(5))+24*p1
. *ek*q(6)**4*q(2)*q(5)+56*p1*ek*q(6)**2*q(2)*q(5)+40*p1*ek*q(2)*q(5)+16*p1**2
. *ek*q(6)**2*q(4)**2+ek**2*q(4)**2
T1=18*ek*q(6)**4*q(7)**2+48*ek*q(6)**2*q(7)**2+24*ek*q(7)**2+48*p1**2*q(7)
. **2+18*p1*ek*q(6)*q(6)*q(7)+18*ek*q(6)**4*q(5)*q(7)+48*ek*q(6)**2*q(5)*q(7)
. +24*p1**2*q(6)**2*q(5)*q(7)+24*ek*q(5)*q(7)+24*p1**2*q(5)*q(7)-(48*p1**
. 2*ek*q(6)**2*q(3)*q(7))-(84*p1**2*ek*q(3)*q(7))+36*p1*ek*q(6)**2*q(2)*q(7)
. +60*p1*ek*q(2)*q(7)-(72*p1**3*q(2)*q(7))+8*p1**2*ek*q(6)**2*q(6)**2+8*p1
. **2*ek*q(6)**2+16*p1**4*q(6)**2+24*p1*ek*q(6)*q(5)*q(6)+22*p1**2*ek*q(6)**2
. *q(4)*q(6)+36*p1**2*ek*q(4)*q(6)+8*p1**4*q(4)*q(6)+6*p1**3*ek*q(6)**2*q(
. 1)*q(6)+20*p1**3*ek*q(1)*q(6)-(24*p1**5*q(1)*q(6))+3*ek*q(6)**4*q(5)**2
. +8*ek*q(6)**2*q(5)**2+4*ek*q(5)**2-(40*p1**2*ek*q(6)**2*q(3)*q(5))-(56*p1
. **2*ek*q(3)*q(5))+32*p1*ek*q(6)**2*q(2)*q(5)-(16*p1**3*q(6)**2*q(2)*q(5))
. +40*p1*ek*q(2)*q(5)-(24*p1**3*q(2)*q(5))+14*p1**2*ek*q(6)**2*q(4)**2+25*
. p1**2*ek*q(4)**2
T0=2*p1**2*ek**4*te*(3*(q(7)+q(5))**2+p1**2*(q(6)+q(4))**2)+he
. *(T0+36*p1**2*ek*q(4)**2-(8*p1**3*ek*q(6)**2*q(1)*q(4))+2*p1*ek**2*q(1)*
. q(4)+16*p1**3*ek*q(1)*q(4)+7*p1**2*ek**2*q(3)**2+48*p1**4*ek*q(3)**2+16
. *p1**3*ek*q(6)**2*q(2)*q(3)-(14*p1*ek**2*q(2)*q(3))+8*p1**3*ek*q(2)*q(3)-

```

```

. (20*p1**2*ek*q(6)**2*q(2)**2)+7*ek**2*q(2)**2-(44*p1**2*ek*q(2)**2)+36*p1
. **4*q(2)**2-(12*p1**4*ek*q(6)**2*q(1)**2)+p1**2*ek**2*q(1)**2-(12*p1**4*
. ek*q(1)**2)+12*p1**6*q(1)**2)/6.0+ek*ej*(T1+6*p1**3*ek*q(6)**2*q(1)*
. q(4)+14*p1**3*ek*q(1)*q(4)-(8*p1**5*q(1)*q(4))+16*p1**4*ek*q(3)**2-(4*
. p1**3*ek*q(2)*q(3))-(8*p1**2*ek*q(2)**2)+24*p1**4*q(2)**2-(3*p1**4*ek*q(
. 1)**2)+8*p1**6*q(1)**2)/3.0
T2=36*q(6)**4*q(7)**2+36*q(6)**2*q(7)**2+9*q(7)**2+36*p1*q(6)**3*q(6)*q(7)
. +42*p1*q(6)*q(7)-(96*p1**2*q(6)**2*q(3)*q(7))-(60*p1**2*q(3)*q(7))
. +36*p1*q(6)**4*q(2)*q(7)+78*p1*q(6)**2*q(2)*q(7)+42*p1*q(2)*q(7)+12*p1**2
. *q(6)**2*q(6)**2+3*p1**2*q(6)**2+24*p1**2*q(6)**2*q(4)*q(6)+20*p1**2*q(4)
. )*q(6)-(8*p1*q(6)*q(3)*q(6))-(24*p1**3*q(6)*q(3)*q(6))+12*p1**2*q(6)**3*
. q(2)*q(6)+8*ek*q(6)*q(2)*q(6)-(4*p1**2*q(6)*q(2)*q(6))+12*p1**3*q(6)**2*q(
. 1)*q(6)+14*p1**3*q(1)*q(6)+2*ek*q(6)**2+2*ek*q(4)**2+2*p1**2*q(
. 4)**2+4*p1*ek*q(6)**2*q(1)*q(4)-(24*p1**3*q(6)**2*q(1)*q(4))+4*p1*ek*q(1)*
. q(4)-(4*p1**3*q(1)*q(4))+10*p1**2*ek*q(6)**2*q(3)**2+10*p1**2*ek*q(3)**2
. +32*p1**4*q(3)**2-(20*p1*ek*q(6)**2*q(2)*q(3))+16*p1**3*q(6)**2*q(2)*q(3)
. -(20*p1*ek*q(2)*q(3))-(4*p1**3*q(2)*q(3))+6*p1**2*q(6)**4*q(2)**2+10*ek*
. q(6)**2*q(2)**2
T0=T0+fe*(T2-(22*p1**2*q(6)**2*q(2)**2)+10*ek*q(2)**2-(19*p1**2*q(2
. )**2)+2*p1**2*ek*q(6)**2*q(1)**2-(18*p1**4*q(6)**2*q(1)**2)+2*p1**2*ek*q(1
. )**2-(9*p1**4*q(1)**2))/6.0
T0=T0+2.0/3.0*ek**2*el*(6*ek*q(6)**2*q(7)**2+6*ek*q(7)**2+48*p1**2*q
. (7)**2+24*ek*q(6)**2*q(5)*q(7)+36*p1**2*q(6)**2*q(5)*q(7)+9*ek*q(5)*q(7)+
. 48*p1**2*q(5)*q(7)-(12*p1**2*ek*q(3)*q(7))+9*p1*ek*q(2)*q(7)-(48*p1**3*
. q(2)*q(7))+p1**2*ek*q(6)**2*q(6)**2+2*p1**2*ek*q(6)**2+16*p1**4*q(6)**2+
. 2*p1**2*ek*q(6)**2*q(4)*q(6)+7*p1**2*ek*q(4)*q(6)+16*p1**4*q(4)*q(6)+3*p1
. **3*ek*q(1)*q(6)-(16*p1**5*q(1)*q(6))+6*p1**2*q(6)**4*q(5)**2+3*ek*q(6)**2
. *2*q(5)**2+12*p1**2*q(6)**2*q(5)**2+3*ek*q(5)**2+8*p1**2*q(5)**2-(12*p1
. **2*ek*q(3)*q(5))-(12*p1**3*q(6)**2*q(2)*q(5))+9*p1*ek*q(2)*q(5)-(32*p1
. **3*q(2)*q(5))+p1**2*ek*q(6)**2*q(4)**2+5*p1**2*ek*q(4)**2+2*p1**4*q(4)
. **2+3*p1**3*ek*q(1)*q(4)-(12*p1**5*q(1)*q(4))+8*p1**4*q(2)**2+2*p1**6*
. q(1)**2)
bmn2(7,7)=T0+ek**3*re*(9*ek*q(7)**2+96*p1**2*q(7)**2+48*p1**2*q(6)**2
. *2*q(5)*q(7)+18*ek*q(5)*q(7)+144*p1**2*q(5)*q(7)-(48*p1**3*q(2)*q(7))
. +3*p1**2*ek*q(6)**2+32*p1**4*q(6)**2+6*p1**2*ek*q(4)*q(6)+48*p1**4*q(4)
. *q(6)-(16*p1**5*q(1)*q(6))+32*p1**2*q(6)**2*q(5)**2+9*ek*q(5)**2+48*p1
. **2*q(5)**2-(48*p1**3*q(2)*q(5))+3*p1**2*ek*q(4)**2+16*p1**4*q(4)**2-
. (16*p1**5*q(1)*q(4))/6.0+de*(6*q(6)**2*q(6)**2-(12*p1*q(6)**3*q(3)
. )*q(6))-(12*p1*q(6)*q(3)*q(6))+12*q(6)**3*q(2)*q(6)+12*q(6)*q(2)*q(6)+2*
. q(6)**2*q(4)**2+q(4)**2+4*p1*q(6)**2*q(1)*q(4)+2*p1*q(1)*q(4)+6*p1**2*q(6)
. **4*q(3)**2+14*p1**2*q(6)**2*q(3)**2+7*p1**2*q(3)**2-(12*p1*q(6)**4*q(2)
. *q(3))-(28*p1*q(6)**2*q(2)*q(3))-(14*p1*q(2)*q(3))+6*q(6)**4*q(2)**2+14
. *q(6)**2*q(2)**2+7*q(2)**2+p1**2*(2*q(6)**2+1)*q(1)**2)/6.0
c
do 100 ii=1,7
do 100 jj=ii,7
100 bmn2(jj,ii)=bmn2(ii,jj)
return
end
c
c
c subroutine sbeamk(bmk,ek1)
c
implicit double precision (a-h,o-z)
c
common/elas/ae,de,fe,he,ej,el,re,te,as,ds,fs
dimension bmk(7,7)
do 10 ii=1,7
do 10 jj=1,7
10 bmk(jj,ii)=0.0d0
c
bmk(2,2)=ae
c
bmk(5,5)=he*ek1**2
c
bmk(5,7)=he*ek1**2+fe*ek1
c
bmk(7,7)=he*ek1**2+fe*2*ek1+de
c
bmk(4,4)=9*fs*ek1**2+6*ds*ek1+as
c

```

```

c
bmk(4,6)=9*fs*ekl**2+6*ds*ekl+as
c
bmk(6,6)=9*fs*ekl**2+6*ds*ekl+as
c
do 100 ii=1,7
do 100 jj=ii,7
100 bmk(jj,ii)=bmk(ii,jj)
return
end
c
c
c subroutine bmn1(q,bmn1,ek)
c
Note that kl appears as 'ek' in this subroutine
c
The equations in this subroutine were generated by MACSYMA.
c
implicit double precision (a-h,o-z)
c
common/elas/ae,ds,fe,he,ej,el,re,te,as,ds,fs
dimension bmn1(7,7),q(7)
c
bmn1(2,2)=3*q(2)*ae
bmn1(2,4)=ae*q(4)
bmn1(2,5)=3*ek**2*he*(q(7)+q(5))+ek*(2*q(6)**2+3)*fe*q(7)
bmn1(2,7)=ek*(2*q(6)**2+3)*fe*(2*q(7)+q(5))+3*ek**2*he*(q(7)+q
(5))+(4*q(6)**2+3)*ds*q(7)
bmn1(4,4)=q(2)*ae
bmn1(5,5)=3*ek**2*q(2)*he
bmn1(5,6)=ek*q(6)*fe*q(7)
bmn1(5,7)=ek*fe*(q(6)*q(6)+2*q(6)**2*q(2)+3*q(2))+3*ek**2*q(2)*he
bmn1(6,7)=ek*q(6)*fe*(2*q(7)+q(5))+2*q(6)*ds*q(7)
bmn1(7,7)=ds*(2*q(6)*q(6)+4*q(6)**2*q(2)+3*q(2))+2*ek*fe*(q(6)*q
(6)+2*q(6)**2*q(2)+3*q(2))+3*ek**2*q(2)*he
c
do 100 ii=1,7
do 100 jj=ii,7
100 bmn1(jj,ii)=bmn1(ii,jj)
return
end
c
c
c subroutine bmn2(q,bmn2,ek)
c
note that kl appears as 'ek' in this subroutine
c
the equations in this subroutine were generated by macsyma.
c
implicit double precision (a-h,o-z)
c
common/elas/ae,ds,fe,he,ej,el,re,te,as,ds,fs
dimension bmn2(7,7),q(7)
c
bmn2(2,2)=7.0/6.0*ek**2*he*(q(7)+q(5))**2+(6*q(6)**4+14*q(6)**2+7)*
. *ds*q(7)**2/6.0+5.0/3.0*ek*(q(6)**2+1)*fe*q(7)*(q(7)+q(5))+ae
. *(q(4)**2+3*q(2)**2)/2.0
bmn2(2,4)=q(2)*ae*q(4)
bmn2(2,5)=7.0/3.0*ek**2*q(2)*he*(q(7)+q(5))+ek*fe*(2*q(6)*q(6)
. +5*q(6)**2*q(2)+5*q(2))*q(7)/3.0
bmn2(2,6)=q(6)*(q(6)**2+1)*ds*q(7)**2+2.0/3.0*ek*q(6)*fe*q(7)*(q
. (7)+q(5))
bmn2(2,7)=ek*fe*(2*q(6)*q(6)+5*q(6)**2*q(2)+5*q(2))*(2*q(7)+q(5))/
. 3.0+7.0/3.0*ek**2*q(2)*he*(q(7)+q(5))+ds*(6*q(6)**3*q(6)+6*
. q(6)*q(6)+6*q(6)**4*q(2)+14*q(6)**2*q(2)+7*q(2))*q(7)/3.0
bmn2(4,4)=ek**2*he*(q(7)+q(5))**2/6.0+(2*q(6)**2+1)*ds*q(7)
. **2/6.0+ek*(q(6)**2+1)*fe*q(7)*(q(7)+q(5))/3.0+ae*(3*q(4)**2
. +q(2)**2)/2.0
bmn2(4,5)=ek**2*he*q(4)*(q(7)+q(5))/3.0+(ek*q(6)**2+ek)*fe*q(4)
. *q(7)/3.0

```

```

bmn2(4,7)=ek*(q(6)**2+1)*fe*q(4)*(2*q(7)+q(5))/3.0+ek**2*he*q(
. 4)*(q(7)+q(5))/3.0+(2*q(6)**2+1)*de*q(4)*q(7)/3.0
bmn2(5,5)=3.0/2.0*ek**4*re*(q(7)+q(5))**2+ek**2*(q(6)**2+2)*(3*q(6)
. **2+2)*ej*q(7)**2/3.0+2*ek**3*(q(6)**2+1)*el*q(7)*(q(7)+q(5)
. )+ek**2*he*(q(4)**2+7*q(2)**2)/6.0
bmn2(5,6)=2.0/3.0*ek*q(6)*q(2)*fe*q(7)
bmn2(5,7)=ek*he*(6*(q(6)**2+1)*(2*q(6)**2+1)*q(7)**2+ek*(q(4)**2+7*q
. (2)**2))/6.0+ek**2*(q(6)**2+2)*(3*q(6)**2+2)*ej*q(7)*(3*q(7)+2*q(5)
. )/3.0+ek**3*(q(6)**2+1)*el*(q(7)+q(5))*(3*q(7)+q(5))+3.0/2.0*ek
. **4*re*(q(7)+q(5))**2+ek*fe*(4*q(6)*q(2)*q(6)+q(6)**2*q(4)**2+
. q(4)**2+5*(q(6)**2+1)*q(2)**2)/6.0
bmn2(6,6)=q(6)**2*de*q(7)**2
bmn2(6,7)=2.0/3.0*ek*q(6)*q(2)*fe*(2*q(7)+q(5))+2*q(6)*de*(q(6)*
. q(6)+q(6)**2*q(2)+q(2))*q(7)
bmn2(7,7)=fe*(9*(2*q(6)**2+1)**2*q(7)**2+2*ek*(4*q(6)*q(2)*q(6)+q(6)
. **2*q(4)**2+2*q(4)**2+5*(q(6)**2+1)*q(2)**2)/6.0+ek**2*(q(6)**2+2)*(3*
. q(6)**2+2)*ej*(6*q(7)**2+6*q(5)*q(7)+q(5)**2)/3.0+ek*he*(12*
. (q(6)**2+1)*(2*q(6)**2+1)*q(7)*(2*q(7)+q(5))+ek*(q(4)**2+7*q(2)**2))/
. 6.0+2*ek**3*(q(6)**2+1)*el*(q(7)+q(5))*(2*q(7)+q(5))+3.0/2.0*ek**
. 4*re*(q(7)+q(5))**2+de*(6*q(6)**2*q(6)**2+12*q(6)**3*q(2)*q(
. 6)+12*q(6)*q(2)*q(6)+2*q(6)**2*q(4)**2+q(4)**2+(6*q(6)**4+14*q(6)**2+7)*q
. (2)**2)/6.0
c
do 100 ii=1,7
do 100 jj=ii,7
100 bmn2(jj,ii)=bmn2(ii,jj)
return
end
.

```

Bibliography

1. Adeli, Hojjat, M.P. Kamat, Girish Kulkarni, and R.D. Vanluchene. "High-Performance Computing in Structural Mechanics and Engineering," Journal of Aerospace Engineering, 6: 249-267, (July 1993).
2. Agarwal, Bhagwan D. and Lawrence J. Broutman. Analysis and Performance of Fiber Composites. New York: John Wiley& Sons, Inc., 1990.
3. Antman, S.S. "Nonlinear Problems of Geometrically Exact Shell Theories," Analytical and Computational Models of Shells, 109-131 (1989).
4. Belytschko, Ted and Lawrence W. Glaum. "Applications of Higher Order Corotational Stretch Theories to Nonlinear Finite Element Analysis," Computers and Structures, 10: 175-182 (1979).
5. Brockman, Robert A. Magna: A Finite Element Program for the Materially and Geometrically Nonlinear Analysis of Three-Dimensional Structures Subjected to Static and Transient Loading. Technical Report (January 1981).
6. Chandrashekara, K. "General Nonlinear Bending Analysis of Composite Beams, Plates and Shells," Proceedings of International Conference on Composite Materials and Structures, 392-402 (1988).
7. Cook, Robert D., David S. Malkus and Michael E. Plesha. Concepts and Applications of Finite Element Analysis. New York: John Wiley& Sons, 1989.
8. Creaghan, Stephen G. Nonlinear Large Displacement and Moderate Rotational Characteristics of Composite Beams Incorporating Transverse Shear Strain. Masters Thesis, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1992.
9. Crisfield, M.A. "A Fast Incremental/Iterative Solution Procedure That Handles Snap Through," Computers and Structures, 13: 55-62 (1981).
10. DaDeppo, D.A. and R. Schmidt. "Large Deflections and Stability of Hingless Circular Arches Under Interacting Loads," Journal of Applied Mechanics, 989-994 (December 1974).
11. DaDeppo, D.A. and R. Schmidt. "Instability of Clamped-Hinged Circular Arches Subjected to a Point Load," Transactions of ASME, 42:894-896 (December 1975).

12. Epstein, Marcello and David W. Murray. "Large Deformation In-Plane Analysis of Elastic Beams," Computers and Structures, 6:1-9 (1976).
13. Farhat, Charbel, Edward Wilson and Graham Powell. "Solution of Finite Element Systems on Concurrent Processing Computers," Engineering with Computers, 2: 157-165 (1987).
14. Fung, Y.C. and E.E. Sechler. Thin-Shell Structures Theory Experiment and Design. New York: Prentice-Hall Inc., 1974.
15. Hsiao, Kou Mo and Fang Yu Hou. "Nonlinear Finite Element Analysis of Elastic Frames," Computers and Structures, 26:693-701 (1987).
16. Huang, Nai-Chien. "Unsymmetrical Buckling of Thin Shallow Spherical Shells," Journal of Applied Mechanics, 447-457 (September 1964).
17. Huddelston, J.V. "Finite Deflections and Snap-Through of High Circular Arches," Journal of Applied Mechanics, 763-769 (December 1968).
18. Karamanlidis, D., A. Honecker and K. Knothe. "Large Deflection Finite Element Analysis of Pre- and Postcritical Response of Thin Elastic Frames," Nonlinear Finite Element Analysis in Structural Mechanics edited by W. Wunderlich, et al., New York: Springer-Verlag, 1981.
19. Koiter, W.T. "A Consistent First Approximation in the General Theory of Thin Elastic Shells," Computers and Structures, 27:No5, 619-623 (1987).
20. Kreyzig, Erwin. Advanced Engineering Mathematics. New York: John Wiley&Sons, 1988.
21. Mindlin, R.D. "Influence of Rotary Inertia and Shear on Flexural Motions of Isotropic, Elastic Plates," Journal of Applied Mechanics, 18:31-38 (March 1951).
22. Minguet, Pierre and John Dugundji. "Experiments and Analysis for Composite Blades Under Large Deflections Part I: Static Behavior," AIAA Journal, 28:1573-1579 (September 1990).
23. Mondkar, D.P. and G.H. Powell. "Finite Element Analysis of Non-Linear Static and Dynamic Response," International Journal for Numerical Methods in Engineering, 11:499-520 (1977).
24. Noguchi, H. and T. Hisada. "Sensitivity Analysis in Post-Buckling Problems of Shell Structures," Computers and Structures, 47:699-710 (1993).

25. Nolte, L.P., J. Makowski and H. Stumpf. "On the Derivation and Comparative Analysis of Large Rotation Shell Theories," Ingenieur-Archiv, **56**:145-160 (1986).
26. Nygard, M.K. and P.G. Bergan. "Advances in Treating Large Rotations for Nonlinear Problems," State-of-the-Art Surveys on Computational Mechanics, 305-333 (1989).
27. Palazotto, Anthony N. and Scott T. Dennis. Nonlinear Analysis of Shell Structures. Washington D.C.: American Institute of Aeronautics and Astronautics, 1992.
28. Reddy, J.N. "A Simple Higher-Order Theory for Laminated Composite Plates," Journal of Applied Mechanics, **51**:745-752 (December 1984).
29. Reissner, Eric. "The Effect of Transverse Shear Deformation on the Bending of Elastic Plates," Journal of Applied Mechanics, A-69-A-77 (June 1965).
30. Saada, Adel S. Elasticity Theory and Applications. Malabar, FL: Krieger, 1989.
31. Sabir, A.B. and A.C. Lock. "Large Deflection, Geometrically Non-Linear Finite Element Analysis of Circular Arches," International Journal of Mechanical Sciences, **15**:37-47 (1973).
32. Schmidt, R. and D.A. DaDeppo. "Nonlinear Theory of Arches with Transverse Shear Deformation and Rotary Inertia," Industrial Mathematics, **21**:33-49 (1971).
33. Smith, R.A. Higher-Order Thickness Expansions for Cylindrical Shells. PhD Dissertation, School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, 1991.
34. Symbolics Inc. Macsyma Version13. User's Manual. Burlington MA: November 1988.
35. Tsai, C.T. and A.N. Palazotto. "Nonlinear and Multiple Snapping Responses of Cylindrical Panels Comparing Displacement Control and Riks Method," Computers and Structures, **41**:605-610 (1991).
36. VanLuchene, R.D., R.H. Lee and V.J. Meyers. "Large Scale Finite Element Analysis on a Vector Processor," Computers and Structures **24**:625-635 (1986).

Vita

Captain Daniel A Miller II was born on 6 April 1965 in Rhinelander, Wisconsin. He graduated from Libby Senior High School in Libby, Montana in 1983. He then attended Montana State University where he graduated with a Bachelor of Science in Mechanical Engineering. Upon graduation, he received a regular commission in the USAF and spent his first duty assignment at the Ballistic Missile Organization in SanBernadino, California. He was assigned to the Air Force Institute of Technology in May 1992.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED	
	December 1993	Master's Thesis	
4. TITLE AND SUBTITLE		5. FUNDING NUMBERS	
NONLINEAR LARGE DEFORMATION THEORY OF COMPOSITE ARCHES USING TRUNCATED ROTATIONS			
6. AUTHOR(S)			
Daniel A. Miller II, Captain, USAF			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)		8. PERFORMING ORGANIZATION REPORT NUMBER	
Air Force Institute of Technology, WPAFB OH 45433-6583		AFIT/GAE/ENY/93D-22	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES			
12a. DISTRIBUTION / AVAILABILITY STATEMENT		12b. DISTRIBUTION CODE	
Approved for public release; distribution unlimited			
13. ABSTRACT (Maximum 200 words)			
<p>This research has been directed toward capturing large cross sectional rotation during bending of composite arches. A potential energy based nonlinear finite element model that incorporates transverse shear strain was modified to include large bending rotations. Large rotation kinematics were derived in a vector format leading to nonlinear strain that was decomposed into convenient forms for inclusion in the potential energy function. Problem discretization resulted in a finite element model capable of large deformations. Riks method and displacement control solution techniques were used. Numerous problems and examples were compared and analyzed. Code vectorization and parallelization were briefly examined to improve computational efficiency.</p>			
14. SUBJECT TERMS		15. NUMBER OF PAGES 177	
Large Rotation Kinematics, Composite Arches, Parabolic Shear, Nonlinear Finite Element, Truncated Rotations, Total Lagrangian		16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL